ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

IMPLEMENTACIÓN DE EXTREMO A EXTREMO EN WS-BPEL 2.0 DE PROCESOS DE NEGOCIO INTERACTIVOS PARA GESTIÓN INMOBILIARIA

María de los Ángeles Villalba Fernández

1 de julio de 2015

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

IMPLEMENTACIÓN DE EXTREMO A EXTREMO EN WS-BPEL 2.0 DE PROCESOS DE NEGOCIO INTERACTIVOS PARA GESTIÓN INMOBILIARIA

- Departamento: Ingeniería Informática
- Directores del proyecto: Antonia Estero Botaro y Antonio García Domínguez
- Autora del proyecto: María de los Ángeles Villalba Fernández

Cádiz, 1 de julio de 2015

Fdo.: María de los Ángeles Villalba Fernández

Agradecimientos

Quisiera agradecer al grupo de investigación UCASE que haya confiado en mi para llevar a cabo este proyecto, y en especial, a mis tutores Antonia Estero Botaro y Antonio García Domínguez por tutorizar este proyecto, por todos los conocimientos y el tiempo que me han dedicado.

A mi familia, sobre todo a mis padres y mis hermanos por apoyarme, por poder siempre contar con ellos y por darme fuerzas día tras día para continuar y no caer en la tentación de abandonar. Ellos han visto las horas de trabajo y el esfuerzo que le he dedicado a este proyecto. Por todo lo que me habéis ayudado y soportado, gracias.

A mis amigos, sobre todo a los más íntimos, que han compartido conmigo este camino y han vivido todo ese esfuerzo dedicado, los que han estado en las malos y buenos momentos. Gracias por animarme a seguir adelante con el proyecto y confiar en mí.

Por último, aunque no menos importante, a los familiares que ya no están entre nosotros, que confiaban en mí y me hubiera gustado que vieran como lo he conseguido.

Índice general

Índice de figuras

Índice de tablas

1. Motivación y contexto

A lo largo de este primer capítulo se describirá la motivación para llevar a cabo la realización del proyecto mediante una introducción, además de los objetivos que se han de cumplir y una descripción general del mismo. También, se definen algunos conceptos básicos y las distintas tecnologías que han sido utilizadas para desarrollar el proyecto.

1.1. Introducción

Este Proyecto Fin de Carrera (PFC) se enmarca como trabajo de colaboración dentro del grupo de investigación UCASE del Departamento de Ingeniería Informática de la Universidad de Cádiz. Las líneas de investigación de este grupo comprenden actualmente la ingeniería de servicios, arquitecturas dirigidas por eventos, arquitecturas orientadas a servicios, desarrollo dirigido por modelos y verificación y validación de software.

El grupo de investigación UCASE trabaja en una línea de investigación sobre pruebas de composiciones de servicios web en Web Services Business Process Execution Language (WS-BPEL) [3], y hasta ahora sólo han contado con composiciones artificiales en las que los servicios invocados son simulados por el marco de pruebas unitarias BPELUnit [4].

El objetivo del PFC es desarrollar un caso de estudio realista en el que se invoque a servicios web reales con su propia lógica de negocio y capa de persistencia, y en el que los usuarios realicen algunas tareas de forma manual. Composiciones de este tipo ayudaría a la validación de la investigación del grupo UCASE.

La realización de este PFC permite adquirir nuevos conocimientos bastante útiles y necesarios en el ámbito laboral, destacando los lenguajes WS-BPEL 2.0 y Java [5, 6] o los servicios web, entre otros.

A través de Daniel Lübke, consultor de la empresa suiza innoQ GmbH, se obtuvo un caso de estudio basado en la documentación de un proyecto de gobierno electrónico suizo [7, 8]. Este PFC gestiona los asuntos relativos a la propiedad inmobiliaria y los créditos hipotecarios. Dicho caso de estudio trata sobre Terravis, es una plataforma constituida por procesos que permite la gestión de algunos negocios entre profesionales, como por ejemplo: las oficinas del registro de la propiedad, los notarios y las entidades de crédito (los bancos, los seguros y los fondos de pensiones). En Suiza, los diferentes profesionales tratan asuntos como la creación de cédulas hipotecarias, cambiar el banco acreedor de nuestra hipoteca, venta de inmuebles, etc. Estas operaciones se realizan en papel y por correo electrónico y por lo tanto, Terravis va a permitir realizar ciertas gestiones de forma electrónica.

Terravis sólo se centra en las transacciones inmobiliarias más comunes, no va a tratar casos especiales. Los propietarios no tendrán acceso, se seguirá visitando a la entidad de crédito para las solicitudes de crédito y al notario para los asuntos relativos al registro de la propiedad. Los procesos que no son llevados a cabo a través de Terravis, se escriben en papel y se envía por correo el formulario que se utilizará. No hay ninguna obligación de utilizar Terravis, sin embargo, las eficiencias son evidentes.

Este caso de estudio se ha simplificado para adaptarlo a este proyecto y en base a esto, se ha implementado nuestro PFC desarrollando un sistema orientado a servicios con procesos de negocio de gestión inmobiliaria. A continuación, se nombran los procesos que en este PFC se han desarrollado:

- Registrar cédula hipotecaria.

- Cambio de acreedor / Recompra de créditos entre entidades de crédito.
- Constitución de cédulas hipotecarias de registro.

1.2. Objetivos

Como se ha comentado en el inicio de este trabajo, el objetivo principal de este PFC es desarrollar un caso de estudio realista en el que se invoque a servicios web reales con su propia lógica de negocio y capa de persistencia, y en el que los usuarios realicen algunas tareas de forma manual. Para lograr este objetivo se ha subdividido en tres subobjetivos y cada subobjetivo requiere unas tareas concretas:

1. **La capa de persistencia y de dominio.**
   - Elección de la herramienta y lenguaje para la creación de la capa de persistencia y de dominio.
   - Implementación de la capa de persistencia y de dominio de las composiciones. He utilizado la plataforma de programación Java Platform Enterprise Edition (Java EE) para crear un proyecto web Java estándar con Maven [9, 10], el cuál, me permitió crear un proyecto multimódulo compuesto por la capa de persistencia, de dominio y los servicios web.
   - Realización de los casos de prueba para la capa de persistencia y de dominio.
2. **Los servicios web.**
   - Implementación de los servicios web necesarios en cada una de las composiciones a realizar.
   - Realización de pruebas para los servicios web.
3. **Las composiciones.**
   - Elección de la herramienta gráfica a utilizar para realizar las composiciones. Se pueden usar diversos editores gráficos para la implementación de las composiciones WS-BPEL y así, facilita no tener que escribir el código WS-BPEL directamente mediante un editor de texto. Además, facilita la creación de otros ficheros necesarios que forman parte de la composición.
   - Desarrollo de cada una de las composiciones de servicios web en WS-BPEL 2.0. Como se ha nombrado anteriormente dichas composiciones deben ir junto con un conjunto de ficheros complementarios.
     - Fichero con extensión BPEL. Se implementa cada composición, define el proceso de negocio.
     - Fichero con extensión WSDL. Se implementa la especificación de los distintos servicios con los que se comunica el proceso Business Process Execution Language (BPEL).
     - Se debe disponer de un conjunto de casos de prueba para la ejecución de cada composición WS-BPEL. Los ficheros de casos de prueba tienen un formato especial basado en eXtensible Markup Language (XML) y tienen extensión BPELUnit Test Suite (BPTS).
   - Finalmente, las composiciones realizadas deben de cumplir una serie de requisitos, si los cumplen, deben añadirse en un repositorio de composiciones que mantiene el grupo de investigación UCASE.

Para el desarrollo de este PFC ha sido necesario aprender los lenguajes WS-BPEL 2.0 y Java. Además, Web Services Description Language (WSDL) [11] para la descripción de los servicios web, BPELUnit para la realización de las pruebas unitarias en las composiciones, DbUnit [12] para la realización de las pruebas unitarias en la capa de persistencia, la notación BPMN [13, 1], para el modelado de procesos de negocio, Maven, XML Schema [14], Simple Object Access Protocol (SOAP) [15], XPath [16] y XQuery [17].

1.3. Restricciones generales

Redmine [18] es una herramienta para la gestión de proyectos que incluye un sistema de seguimiento de incidentes con seguimiento de errores. Dicha herramienta es utilizada por el grupo UCASE para la gestión de sus proyectos y el contenido de este PFC está subido aquí. Otras herramientas que incluye son calendario de actividades, diagramas de Gantt para la representación visual de la línea del tiempo de los proyectos, wiki, foro, visor del repositorio de control de versiones, Really Simple Syndication (RSS), control de flujo de trabajo basado en roles, fuentes web y notificaciones por correo electrónico, administración de noticias, documentos y archivos, Integración Software Configuration Management (SCM) (Subversion, Concurrent Versions System (CVS), Git, etc), soporta diferentes bases de datos (MySQL, PostgreSQL y SQLite), etcétera. Está escrito usando el «framework» Ruby on Rails. Es software libre y de código abierto, disponible bajo la Licencia Pública General de GNU v2.

Las composiciones WS-BPEL creadas deben cumplir una serie de restricciones para poder subirse al repositorio que utiliza el grupo de investigación UCASE.

Para este proyecto se ha usado Subversion (SVN) [19], es un software libre de control de versiones o Version control systems (VCS) con licencia Apache/Berkeley Software Distribution (BSD), diseñado para reemplazar a CVS. También conocido como SVN, nombre usado en la línea de órdenes. Una diferencias con CVS es que subversion todo el proyecto está bajo el mismo número de revisión; mientras que CVS cada archivo tiene un número de revisión independiente.

Subversion maneja ficheros y directorios, y los cambios hechos a ellos, con el tiempo. Esto le permite recuperar versiones antiguas de sus datos o examinar el historial de cómo cambiaron sus datos. Permite que pueda ser usado por personas en diferentes equipos. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración, y la calidad no se resiente, ya que si algún cambio es incorrecto siempre se puede deshacer dicho cambio en el proyecto, al disponer de todas las versiones previas del mismo.

Algunos sistemas de control de versiones también son sistemas de gestión de configuración de software (SCM). Estos sistemas están diseñados específicamente para manejar árboles de código fuente, y tienen muchas características que son específicas para el desarrollo de software. Subversion, sin embargo, no es uno de estos sistemas. Es un sistema general que puede ser utilizado para manejar cualquier colección de archivos.

Algunas ventajas de Subversion:

- Se mantiene el histórico de los archivos y directorios a través de copias y renombramientos.
- No se tendrán que subir archivos a la forja uno a uno, todos los cambios que afecten a uno o varios archivos, subidos de una sola vez cuentan como una modificación.
- La creación de ramas y etiquetas es más eficiente que en CVS.
- Con CVS se envían al servidor los ficheros completos, mientras que, con SVN sólo se mandan los cambios que han sufrido los ficheros.
- SVN maneja los ficheros binarios eficientemente.
- Permite selectivamente el bloqueo de archivos para que no sean editados por más de una persona al mismo tiempo.
- Se puede usar las opciones que Apache posee para autentificar archivos, cuando se integra con él .

Entre las carencias, se puede destacar, el manejo de cambio de nombres de archivos no es completo. Lo maneja como la suma de una operación de copia y una de borrado.

Para integrar el sistema de SVN de control de versiones con la plataforma Eclipse [20] he utilizado Subversive [21]. Usando el «plug-in» de Subversive, se puede trabajar con proyectos almacenados en los repositorios de Subversion directamente desde Eclipse workbench de una manera similar a trabajar con otros proveedores de control de versiones de Eclipse, como CVS y Git.

Subversion es elegido porque se puede aprender fácilmente por los miembros del grupo UCASE. Además, al estar subido a la forja, no se pierde nuestro proyecto y ha sido útil para deshacer cambios. Para más información sobre este sistema de control de versiones, se puede encontrar en el manual de Subversion.

1.4. Conceptos básicos

Aquí se detallan una serie de conceptos básicos y las distintas tecnologías que han sido utilizadas para desarrollar el proyecto.

1.4.1. El lenguaje Java

La capa de persistencia, de dominio y los servicios web de este proyecto han sido desarrollados en Java, porque su uso presenta muchas ventajas y por lograr integración con el resto de sistemas con los que se comunica. Es el lenguaje que se ha usado por el grupo UCASE para el desarrollo de todas las herramientas relacionadas con la investigación de las pruebas de mutación. Java [5, 6] es un lenguaje de programación orientado a objetos, desarrollado por «Green Team», un grupo de trece personas, dirigido por «James Gosling», para la empresa «Sun Microsystems» en el año 1991. Los objetivos de «Gosling» eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++.

Se denominó inicialmente «Oak» (roble en inglés, según la versión más aceptada, por el roble que veía a través de la ventana de su despacho). El resultado fue un lenguaje que tenía similitudes con C, C++ y «Objective C» y que no estaba ligado a un tipo de Central Processing Unit (CPU) concreto. Más tarde, se cambiaría el nombre de «Oak» a Java, por cuestiones de propiedad intelectual, al existir ya un lenguaje con el nombre de «Oak». Se supone que le pusieron ese nombre mientras tomaban café (Java es nombre de un tipo de café, originario de Asia), aunque otros afirman que el nombre deriva de las siglas de «James Gosling», «Arthur Van Hoff», y «Andy Bechtolsheim».

La compañía «Sun» describe el lenguaje Java como simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico. Entre las razones que hacen tan atractiva la orientación a objetos figuran:

- La relativa cercanía de sus conceptos a las entidades que aparecen en el mundo real.
- La simplicidad del modelo, que emplea los mismos elementos fundamentales para expresar de manera uniforme el análisis, el diseño y la implementación de un sistema.
- La gran capacidad de adaptación e integración de sus modelos, que facilita la realización de modificaciones, incluso durante el proceso de desarrollo, y el mantenimiento.
- La posibilidad de aumentar las oportunidades de reutilización de componentes de software en proyectos distintos.

El uso de las técnicas de la Programación Orientada a Objetos (POO) puede reducir fácilmente un gran conjunto de problemas a una sencilla solución. Java no utiliza punteros, sino que los objetos son accedidos mediante referencias. Además, es un lenguaje interpretado, el compilador Java traduce cada fichero fuente de clases a código de «bytes» («Bytecode») que se envía al visualizador solicitado y se interpreta en la máquina

que posee un interprete de Java o dispone de un visualizador que funciona con Java. Java no permite la carga de operadores y tiene un sistema automático de asignación y liberación de memoria (recolector de basura).

1.4.2. El lenguaje WS-BPEL 2.0

Introducción.

WS-BPEL [3] es un acrónimo de «Web Services Business Process Execution Language». WS-BPEL es un lenguaje formal para expresar un proceso de negocio como una composición de servicios web. WS-BPEL 2.0 fue elegido estándar oficial por Organization for the Advancement of Structured Information Standards (OASIS) en Abril de 2007. Estos servicios se especifican mediante el lenguaje WSDL, a través de este lenguaje se pueden definir los mensajes mediante los que pueden interactuar los distintos servicios web durante su comunicación, así como, la forma de establecer la comunicación entre los distintos servicios web y los protocolos de comunicación usados.

Estructura básica de un Proceso de Negocio.

WS-BPEL es un lenguaje basado en XML. Su espacio de nombres es:

`http://docs.oasis-open.org/wsbpel/2.0/process/executable`

El elemento raíz es `<process>`, y debe definirse, como mínimo, el atributo `name`: el nombre del proceso. Adicionalmente pueden especificarse los atributos siguientes:

- **`queryLanguage`.** Este atributo especifica el lenguaje de consulta utilizado en el proceso de selección de los nodos de asignación. El valor por defecto de este atributo es: "Urn: oasis: names: tc: WSBPEL: 2.0: sublang: xpath1.0", que representa el uso de XPath 1.0 dentro de WS-BPEL 2.0.
- **`expressionLanguage`.** Este atributo especifica el lenguaje de expresión utilizada en el `<process>`. El valor por defecto de este atributo es: "Urn: oasis: names: tc: WSBPEL: 2.0: sublang: xpath1.0", que representa el uso de XPath 1.0 dentro de WS-BPEL 2.0.
- **`suppressJoinFailure`.** Este atributo determina si el fallo joinFailure será suprimido para todas las actividades en el proceso. El efecto de este atributo a nivel del proceso puede ser sobrescrito por una actividad usando un valor diferente para el atributo. El valor predeterminado para este atributo es "no" a nivel de proceso.
- **`exitOnStandardFault`.** Si el valor de este atributo se establece en "yes", entonces el proceso debe salir de inmediato, como si se ha llegado a una actividad `<exit>`, cuando ocurra un fallo estándar distinto de bpel:joinFailure. Si el valor de este atributo se establece en "no", el proceso puede manejar un error estándar utilizando un manejador de errores. El valor predeterminado para este atributo es "no".

WS-BPEL está diseñado para ser extensible. Podría añadirse un elemento `<extensions>` al proceso dentro del cual se especifiquen tantas `<extension>` como sea necesario. Se utiliza para declarar espacios de nombres de atributos / elementos de extensión WS-BPEL e indicar si ellos llevan la semántica que debe ser entendida por un procesador WS-BPEL.

El elemento de declaración `<extension>` bajo `<extensions>` es en sí extensible. Estos elementos deben tener dos atributos:

- **`namespaces`:** El espacio de nombres de la extensión.
- **`mustUnderstand`:** Especifica si el motor de ejecución debe entender la extensión, o puede ignorarla.

Además de las extensiones, un proceso puede contener los siguientes elementos en su primer nivel:

- **<import>:** Importación de otros ficheros: declaraciones WSDL, XML Schema Definition (XSD)...
- **<partnerLinks>:** Declaración de los enlaces a otros servicios (Agentes Externos).
- **<messageExchanges>:** Intercambio de mensajes.
- **<variables>:** Declaración de variables globales.
- **<correlationSets>:** Vincula un mensaje con una instancia concreta del proceso.
- **<faultHandlers>:** Manejadores de fallos. El elemento `catch` permite capturar un fallo específico y realizar ciertas actividades que se definan y el elemento `catchAll` intercepta cualquier tipo de fallo.
- **<eventHandlers>:** Manejadores de eventos. Mediante los elementos `onEvent` determinan las acciones a realizar cuando se produce un evento dado, y mediante un elemento `onAlarm` las actividades a realizar, si después de un tiempo determinado no se ha producido ningún evento.
- **Actividades:** La actividad principal del proceso.

Actividades.

Las componentes principales de los procesos BPEL son las actividades. Hay dos tipos:

1. **Actividades básicas:** Son aquellas que describen los pasos elementales del comportamiento del proceso.
   - **<invoke>:** Se utiliza para llamar a servicios web ofrecidos por los proveedores de servicios. El uso típico es invocar una operación de un servicio, que se considera una actividad básica. La actividad `<invoke>` puede incluir otras actividades, entre líneas de manejador de compensación y manejadores de errores. Las operaciones pueden ser operaciones «request-response» o «one-way», correspondiendo a las definiciones de operación WSDL.
   - **<receive>:** Espera la llegada de un mensaje en una operación específica. Esta actividad especifica el `PartnerLink` que contiene el myrole utilizado para recibir mensajes, el `portType` (opcional) y la operación que espera «partner» para invocar.
   - **<reply>:** Devuelve un mensaje en respuesta de uno recibido previamente. Se utiliza para enviar una respuesta a una solicitud previamente aceptada a través de una actividad de mensajes entrantes, tales como la actividad `<receive>`. Estas respuestas sólo son significativas para las interacciones de «request-response».
   - **<assign>:** Se puede utilizar para copiar datos de una variable a otra, así como para construir e insertar nuevos datos mediante expresiones. El uso de las expresiones es motivada principalmente por la necesidad de realizar cálculos simples (tales como incrementar los números de secuencia). Pueden operar en las variables, propiedades, y constantes literales para producir un nuevo valor. También, se puede utilizar para copiar referencias desde y hacia Agentes Externos. También es posible incluir las operaciones de manipulación de datos extensibles definidos como elementos de extensión bajo diferentes espacios de nombres del espacio de nombres WS-BPEL. La actividad `<Assign>` contiene una o más de las operaciones elementales.
   - **<throw>:** Lanza un fallo. Se utiliza cuando un proceso de negocio necesita lanzar un fallo interno explícitamente. La actividad `<throw>` proporciona el nombre del fallo, y puede opcionalmente proporcionar datos con más información sobre el fallo. Un manejador de errores puede utilizar estos datos para manejar el fallo y para rellenar mensajes de fallo que necesitan ser enviados a otros servicios.

- **<wait>:** Provoca la parada del proceso durante un cierto periodo de tiempo o hasta que se alcanza un cierto plazo. Un uso típico de esta actividad es para invocar una operación en un momento determinado.
- **<empty>:** Actividad que no hace nada, pero puede ser útil para la sincronización de actividades concurrentes. A menudo, hay una necesidad de usar una actividad que no hace nada, por ejemplo, cuando un fallo necesita ser atrapado y suprimido. La actividad <empty> se utiliza para este propósito. Otro uso de <empty> es proporcionar un punto de sincronización en un <flow>.
- **<extensionActivity>:** Agrega nuevos tipos de actividad. Una definición de proceso WS-BPEL puede incluir nuevas actividades, que no están definidas por esta especificación, mediante la colocación en el interior del elemento de <extensionActivity>.
- **<exit>:** Se utiliza para poner fin inmediatamente a la instancia del proceso de negocio. Todas las actividades que se están ejecutando actualmente deben terminar de inmediato y sin involucrar un manejador de terminación, de fallos, o comportamiento de compensación.
- **<rethrow>:** Lanza un fallo que ha sido capturado por un manejador de fallos. Se utiliza en manejadores de errores para volver a lanzar el fallo, es decir, el nombre de error y, en la actualidad, los datos de error del error original. Se puede usar sólo dentro de un manejador de errores (<catch> y <catchAll>).
- **<validate>:** Valida una o más variables. Esta actividad se utiliza para validar los valores de las variables con respecto a su definición de datos WSDL y XML asociada.
- **<compensate>:** Se utiliza sólo dentro de un manejador de fallos, de compensación o de terminación para comenzar la compensación de todos los <scope> internos que ya han sido completados con éxito.
- **<compensateScope>:** Se utiliza sólo dentro de un manejador de fallos, de compensación o de terminación para iniciar una compensación de un <scope> específico interno que ya ha sido completado con éxito.

2. **Actividades estructuradas:** Codifican la lógica de flujo de control, y por lo tanto, pueden contener otras actividades básicas y / o estructurados de forma recursiva.
   - **<sequence>:** Contiene una o más de las actividades que se realizan de forma secuencial, en el orden léxico en el que aparecen dentro del elemento <sequence>. La actividad <sequence> completa cuando la última actividad en la secuencia se haya completado.
   - **<if>:** Seleccionar una determinada actividad a ejecutar entre un conjunto de opciones. Proporciona el comportamiento condicional. La actividad consiste en una lista ordenada de una o más ramas condicionales definidos por los elementos <elseif> <if> y opcional, seguido de un elemento <else>. Se toma la primera rama cuya <condition> se evalúa a «true», y se lleva a cabo su actividad correspondiente. Si no se toma ninguna rama con una condición, entonces se toma la rama <else> si existe.
   - **<while>:** Proporciona la ejecución repetida de una actividad contenida. La actividad correspondiente se ejecuta mientras <condition> se evalúa a «true» al comienzo de cada iteración.
   - **<repeatUntil>:** Proporciona la ejecución repetida de una actividad contenida. La actividad correspondiente se ejecuta hasta que <condition> se convierte en realidad. La condición se comprueba después de cada ejecución del cuerpo del bucle. En contraste con la actividad <while>, el bucle <repeatUntil> ejecuta la actividad contenida al menos una vez.
   - **<pick>:** Permite elegir uno de varios eventos (mensajes entrantes o alarmas temporizadas) y continuar por la ruta de ejecución seleccionada. Se trata de una actividad mediante la que,

a través de los elementos <onMessage> permite determinar las actividades a realizar cuando se recibe un determinado mensaje y, a través de un elemento <onAlarm>, especificar las actividades a realizar cuando se alcance un determinado período de tiempo.

- **<flow>:** Proporciona concurrencia y sincronización. Proporciona concurrencia mediante la ejecución de todas sus actividades hijas en paralelo. También, introduce enlaces de sincronización para hacer que una actividad espere por otra, o salte su ejecución por completo.
- **<forEach>:** Es un constructor repetible especial para ejecutar múltiples instancias de la misma actividad <scope>. Ejecutará su actividad <scope> contenida exactamente N +1 veces donde n es igual a <finalCounterValue> menos <startCounterValue>. Esta actividad consta del atributo «parallel», que si toma el valor "yes", se ejecutarían paralelamente las N+1 instancias del <scope> contenido.
- **<scope>:** Permite la declaración de recursos locales, como las variables y los agentes externos y una actividad principal a ejecutar en este entorno local. También, incluye manejadores de compensación, fallos, de terminación y de eventos. Esta actividad tiene el atributo «isolated», cuando dicho atributo toma el valor "yes" se protege el acceso a las variables compartidas, evitando que otro <scope> que utilice las mismas variables pueda acceder a ellas hasta que no haya terminado la ejecución del primer <scope>.

Correlación.

La información proporcionada hasta ahora sugiere que el destino para los mensajes que se entregan a un servicio de procesos de negocio es el puerto WSDL del servicio destinatario. Los mensajes enviados a tales procesos, deben ser entregados no sólo al puerto de destino correcto si no también, a la instancia correcta del proceso de negocio que ofrece el puerto. WS-BPEL usa «tokens» de correlación para proporcionar instancias de enrutamiento de forma automática. Un conjunto de «tokens» de correlación se define como un conjunto de propiedades compartidas por todos los mensajes en el grupo correlacionado. Este conjunto de propiedades se denomina conjunto de correlación. La idea es asignar un nombre a una parte de los mensajes útil para identificar una instancia concreta. Entonces, el sistema asignará a cada instancia el mensaje que tenga el valor asociado a ella. La correlación se utiliza para seguir la pista a las conversaciones entre una instancia particular de un proceso y las correspondientes instancias de sus servicios «partners». Es decir, permite que el «runtime» de BPEL conozca qué instancia del proceso está esperando a sus mensajes «callback». Se utiliza para correlacionar los mensajes de entrada y salida y enrutarlos convenientemente. Los conjuntos de correlación se definen en el proceso BPEL indicando un nombre y una lista de propiedades. Se definen después de las variables.

Los conjuntos de correlación se pueden utilizar en las actividades <receive>, <reply>, <onMessage>, <onEvent> , y <invoke>. Deben iniciarse una única vez en el proceso que inicia la interacción, en una actividad que tenga un mensaje conteniendo las propiedades del conjunto. Las propiedades tienen un nombre, un tipo de dato y una serie de alias (<vprop:propertyAlias>), cada uno identifica a la propiedad dentro de un mensaje concreto definido en el WSDL. Las propiedades se definen en el WSDL.

El atributo «initiate» en <correlation> se utiliza para indicar si se está iniciando el conjunto de correlaciones. Los valores son «yes», «join» y no. En el caso de <invoke>, cuando el operador ejecutan una operación «request/response», se utiliza el atributo de patrón en <correlation> para indicar si la correlación se aplica al mensaje de salida ("«request»"), entrada ("«response»") o ambos ("«request-response»").

Al iniciarse, el motor BPEL mira el valor de las propiedades en el mensaje y se asocia a la instancia con dichos valores. Una vez inicializado, el conjunto puede utilizarse en cualquier actividad que reciba/envíe un mensaje del tipo indicado en uno de los <vprop:propertyAlias>. Al recibir un mensaje, el motor BPEL accede a las propiedades indicadas. El mensaje se enruta al proceso que se corresponde con esos

valores de las propiedades. Al enviar un mensaje, se comprueba que las propiedades del mensaje enviado tienen los valores correctos a la instancia, si no «fault» bpel:correlationViolation.

Este lenguaje es utilizado por el grupo UCASE para esta línea de investigación sobre pruebas de composiciones de servicios web.

1.4.3. WSDL

WSDL 2.0 [11] es un lenguaje propuesto por el World Wide Web Consortium (W3C) basado en XML para describir servicios web. Un documento WSDL proporciona la información necesaria al cliente para interaccionar con el servicio web. Permite describir la interfaz pública de los servicios web; eso significa que detalla los protocolos y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje. WSDL se utiliza a menudo junto con SOAP y XML Schema.

Una de sus ventajas es que permite que se separe la descripción abstracta de la funcionalidad ofrecida por un servicio de detalles concretos de una descripción del servicio, tales como "cómo" y "dónde" se ofrece esa funcionalidad.

Un documento WSDL tiene una estructura semejante a la siguiente:

```
<definitions>
    <types>

    </types>
    <message name=''...''>

    </message>
    <portType name=''...''>

    </portType>
    <binding name=''...''>

    </binding>
    <service name=''...''>

    </service>
</definitions>
```

Por lo tanto, un documento WSDL está divido en dos partes claramente diferenciadas:

1. **Parte abstracta:** Es la parte que define qué hace el servicio a través de los mensajes que envía y recibe.
2. **Parte concreta:** Es la parte que define el "cómo" y "dónde".

La estructura del WSDL tiene una serie de elementos. La parte abstracta tiene:

- **Tipos de datos `<types>`:** El elemento `types` define los tipos de datos usados en los mensajes. Estos tipos de datos pueden construirse con cualquier lenguaje, pero lo más normal es hacerlo con XML Schema.
- **Mensajes `<message>`:** El elemento `message` definine los elementos de mensaje que se van a intercambiar entre el cliente y el servicio web. Cada mensaje puede estar dividido en varias partes. Las partes pueden ser de cualquiera de los tipos definidos anteriormente.

- **Tipos de puerto `<portType>`:** El elemento `portType` define el conjunto de operaciones que soporta el servicio web. Una operación no es más que un grupo de mensajes que serán intercambiados. Los tipos de mensajes que puede haber en una operación son: «input», que define un mensaje de entrada, «output», que define un mensaje de salida, y «fault», que define un mensaje de excepción. WSDL define cuatro tipos de operaciones:

  «Request-response»: La operación puede recibir una solicitud y devolverá una respuesta.

  «One-way»: La operación puede recibir un mensaje pero no devolverá una respuesta.

  «Solicit-response»: La operación puede enviar una solicitud y esperará una respuesta.

  «Notification»: La operación puede enviar un mensaje, pero no va a esperar una respuesta.

La parte concreta tiene:

- **«Bindings» `<binding>`:** El elemento `binding` especifica los protocolos de comunicación usados. Las definiciones «binding» especifican detalles de formatos del mensaje y el protocolo de transporte para una o más interfaces, es decir, define la implementación de los detalles necesarios para acceder al servicio.

- **Servicios `<service>`:** El elemento `service` define un conjunto de puertos relacionados y dirección de los mismos. Un puerto es un extremo concreto de un servicio web al que se hace referencia por una dirección única. Los puertos que se definen en determinado servicio son independientes. A través de los puertos se integra la especificación de un «binding» con la dirección del servicio.

1.4.4. SOAP

SOAP Versión 1.2 [15] es un protocolo ligero destinado al intercambio de información estructurada en un entorno descentralizado y distribuido. Utiliza tecnologías XML para definir un marco extensible de mensajería que proporciona una construcción de mensajes que se pueden intercambiar a través de una variedad de protocolos subyacentes. El marco ha sido diseñado para ser independiente de cualquier modelo de programación y otra de aplicación semántica específicas particulares.

Dos de los principales objetivos de diseño son la simplicidad y extensibilidad. Un mensaje SOAP consiste en los siguientes elementos:

1. **`SOAP Envelope`:** Envoltura SOAP, está en el directorio raíz de un mensaje SOAP, este elemento define los documentos XML como un mensaje SOAP. Define el comienzo y el final de un mensaje.

2. **`SOAP Header`:** Cabecera SOAP («header») opcional, que puede contener información adicional. Este contiene información específica de la aplicación, como la autenticación, transacción y pago de información relatada al mensaje SOAP.

3. **`SOAP Body`:** Cuerpo SOAP («body») contiene el mensaje que será comunicado entre dos aplicaciones. Es un elemento obligatorio que debe contener un mensaje SOAP. Cuando se habla de estilos de este documento, corresponde a cómo el contenido del elemento `<soap:body>` puede ser estructurado. Esto se especifica como el atributo «style» de un elemento `binding` (normalmente) o un elemento `operation`. SOAP soporta dos estilos de interacción diferentes:

   `Document`: El mensaje SOAP se envía como un "documento" en el elemento `<soap:Body>` sin reglas de formato adicionales que tengan que ser consideradas. Esta es la opción por defecto.

   `RPC`: `<soap:Body>` puede contener sólo un elemento que lleva el nombre de la operación, y todos los parámetros deben estar representados como sub-elementos de este elemento contenedor.

   Como SOAP está basado en XML, los datos dentro de un cuerpo SOAP deben tener el formato de algún dialecto XML. Una codificación define cómo codificar los datos reales de la aplicación en un

formato XML. Esto se especifica como el atributo «use» del elemento `<soap:body>`. Una codificación es identificada por un Uniform Resource Identifier (URI) y es extensible, se han especificado como parte del estándar SOAP dos formatos de codificación:

Encoded: El formato de los datos se codifica mediante un conjunto de reglas detalladas en la especificación SOAP. `Encoded` es la opción adecuada cuando no se desean estructuras complejas, ya que todos los tipos pueden ser perfectamente descritos a través del esquema XML.

Literal: Significa que las definiciones de tipos siguen literalmente una definición de esquema XML.

Teniendo en cuenta que, en principio, estilos y codificaciones pueden mezclarse arbitrariamente, se obtienen las siguientes cuatro combinaciones posibles:

document/literal: Un estilo muy común, que, sin duda, se convierte en el estilo dominante de los servicios web.

document/encoded: No se utiliza.

rpc/encoded: Estilo típico de los antiguos servicios web, sigue la idea Remote Procedure Call (RPC).

rpc/literal: Básicamente, este es el estilo document/literal, pero con envolturas adicionales para una operación y sus parámetros.

4. **SOAP Fault:** Se utiliza para llevar información de error dentro de un mensaje SOAP. `Body` contiene este "`Fault`" que es opcional, el cual, guarda los mensajes de error que han sido relacionados con el mensaje SOAP.

A continuación, se observa parte de código de un servicio web como ejemplo:

```
<wsdl:binding name="LugarServiceImplServiceSoapBinding" type="
    tns:LugarServiceImpl">
     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
         http"/>
     <wsdl:operation name="create">
       <soap:operation soapAction="" style="document"/>
       <wsdl:input name="create">
         <soap:body use="literal"/>
       </wsdl:input>
       <wsdl:output name="createResponse">
         <soap:body use="literal"/>
       </wsdl:output>
     </wsdl:operation>
</wsdl:binding>
```

Cada elemento `wsdl:binding` sólo puede hacer referencia a un único protocolo. El primer «binding» es un «binding» basado en el protocolo SOAP. El elemento `wsdl:operation` describe la operación sobre la que se va a aplicar información de «binding». El elemento `wsdl:input` describe el objeto de entrada al que se va a aplicar información de «binding». El elemento `wsdl:output` describe el objeto de respuesta al que se va a aplicar información de «binding».

En el ejemplo se puede ver que el «binding» es de tipo SOAP a través de Hypertext Transfer Protocol (HTTP). El estilo de SOAP en este caso es «document». Se puede ver que cada operación puede definir un SOAP `style` propio, teniendo como posibles valores también RPC y DOCUMENT. En el ejemplo se indica «document». Cada elemento que conforma la operación, `wsdl:input`, `wsdl:output` o `wsdl:fault` definen el elemento `soap:body`. Este elemento especifica como se incluyen los elementos `wsdl:parts`

en el elemento `SOAP Body`. El atributo «use» marca la codificación que se realiza en la información, y puede ser «literal» o «encoded». En el ejemplo, el objeto de entrada de «create» la codificación del «body» se realiza de forma literal.

1.4.5. XML Schema

XML Schema [14] es una especificación usada para describir la estructura de documentos XML. Esta basada en XML. Fue desarrollado por el W3C.

Los documentos esquema (usualmente con extensión .xsd de XSD) son más complejos que las Document Type Definition (DTD), y se concibieron como una alternativa a ellas, intentando superar sus puntos débiles y buscar nuevas capacidades a la hora de definir estructuras para documentos XML. Una DTD es una descripción de la estructura y sintaxis de un documento XML. Las limitaciones de las DTD son que no permite definir tipos de datos, no permite espacio de nombres, están limitadas a contenido textual e imponen restricciones de repetición. El principal aporte de XML Schema es el gran número de tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y «strings».

Los XML Schemas son documentos bien formados que se crean alrededor del concepto tipo y de instancias a estos tipos. Los tipos de elementos pueden ser:

- **Simples:** Contienen información de caracteres (no permiten ni atributos ni elementos).
- **Complejos:** Contienen cualquier combinación de contenido de elementos, información de caracteres y atributos.

A continuación, se muestra parte de código de un servicio web como ejemplo de un elemento complejo con XML Schema:

```
1  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://impl.
       entidad.servicioweb.aplicacion.grode.com/" elementFormDefault="unqualified
       " targetNamespace="http://impl.entidad.servicioweb.aplicacion.grode.com/"
       version="1.0">
2
3   <xs:element name="retrieveById" type="tns:retrieveById"/>
4
5   <xs:element name="retrieveByIdResponse" type="tns:retrieveByIdResponse"/>
6
7   <xs:complexType name="retrieveById">
8       <xs:sequence>
9         <xs:element minOccurs="0" name="codigoPostal" type="xs:int"/>
10      </xs:sequence>
11    </xs:complexType>
12
13    <xs:complexType name="retrieveByIdResponse">
14      <xs:sequence>
15        <xs:element minOccurs="0" name="return" type="tns:lugarDTO"/>
16      </xs:sequence>
17    </xs:complexType>
18
19    <xs:complexType name="lugarDTO">
20      <xs:sequence>
21        <xs:element minOccurs="0" name="codigoPostal" type="xs:int"/>
22        <xs:element minOccurs="0" name="lugar" type="xs:string"/>
23      </xs:sequence>
```

```
24    </xs:complexType>
25
26  </xs:schema>
```

En este ejemplo se observan los diferentes tipos de elementos simples y complejos. Tiene un tipo complejo de nombre “retrieveByIdResponse”, es el elemento que devuelve la función “retrieveById” y está compuesto por otro elemento complejo llamado “lugarDTO”. Finalmente, este último está formado por elementos simples llamados “codigoPostal” y “lugar”, el primero de tipo “int” y el segundo de tipo “String”.

1.4.6. XPath 2.0

XPath 2.0 [16] es un lenguaje, definido por el W3C, de expresión que opera sobre la estructura lógica abstracta de un documento XML, en lugar de su sintaxis externa. XPath soporta funciones y operadores [22] y esta estructura lógica es conocida como el modelo de datos [23]. XPath se utiliza principalmente para seleccionar partes de un documento XML. Para ello, el documento XML se modela como un árbol de nodos. XPath permite que los nodos pueden seleccionar por medio de una ruta de navegación jerárquica a través de la estructura del documento. El modelo de datos proporciona una representación del árbol de documentos XML, así como valores atómicos, tales como números enteros, cadenas y booleanos, y secuencias que pueden contener tanto las referencias a los nodos en un documento XML y valores atómicos. El resultado de una expresión XPath puede ser una selección de nodos de los documentos de entrada, o un valor atómico, o más generalmente, cualquier secuencia permitida por el modelo de datos. A continuación, se muestra un pequeño ejemplo de una expresión XPath:

```
1  <dataset>
2       <Usuario_Telefono telefono="0439603606" numUsuario="1"/>
3       <Usuario_Telefono telefono="0449473030" numUsuario="2"/>
4  </dataset>
```

```
1  /dataset/Usuario_Telefono[@numUsuario="2"]
```

En el que se seleccionan los `Usuario_Telefono` cuyo atributo `numUsuario` tenga el valor elegido que sean hijos de `dataset`. XPath devuelve 1 elemento:

```
1  <Usuario_Telefono numUsuario="2" telefono="0449473030"/>
```

1.4.7. BPMN 2.0

Hace unos años una nueva notación, llamada «Business Process Modeling Notation» (BPMN) fue desarrollada. BPMN [13, 1] está diseñado para cubrir muchos tipos de modelos y permite la creación de procesos de negocio de extremo a extremo. Los elementos estructurales de BPMN permiten que el espectador sea capaz de diferenciar fácilmente entre las secciones de un diagrama de BPMN. Hay tres tipos básicos de submodelos dentro de un modelo de extremo a extremo BPMN: proceso, coreografía y colaboración.

BPMN fue desarrollado especialmente para el modelado de procesos de negocio de acuerdo con Arquitectura Orientada a Servicios (SOA) [2]. En este proyecto, se ha utilizado BPMN para los procesos de modelización. BPMN es la notación más completa para el modelado de procesos hasta ahora. Ha sido desarrollado bajo Object Management Group (OMG).

Los objetivos más importantes han sido:

- **Desarrollar una notación, que será comprensible en todos los niveles:** En el modelado de procesos de negocio diferentes personas están involucradas, desde los usuarios de negocio, analistas de negocio y los propietarios del proceso, a los arquitectos técnicos y desarrolladores. La administración revisa los procesos de negocio a intervalos periódicos. Por lo tanto, el objetivo de BPMN ha sido proporcionar una notación gráfica fácil de entender, pero lo suficientemente potente para modelar los procesos de negocio a nivel de detalle requerido.
- **Para activar la transformación automática en código ejecutable, es decir, BPEL, y viceversa:** La brecha entre los modelos de procesos de negocio y la tecnología de la información (software de aplicación) ha sido muy importante en las tecnologías existentes. No existe una definición clara de cómo uno se relaciona con el otro. Por lo tanto, BPMN ha sido diseñado específicamente para proporcionar este tipo de transformaciones.

BPMN 2.0 se puede asignar a más de una plataforma de procesos dependientes de lenguaje de modelado, por ejemplo, WS-BPEL 2.0. Este documento incluye un mapeo de un subconjunto de BPMN para WS-BPEL 2.0. La especificación utiliza otras normas para la definición de tipos de datos, expresiones y operaciones de servicio. Estas normas son de esquemas XML, XPath y WSDL, respectivamente.

Debe hacerse hincapié en que uno de los controladores para el desarrollo de BPMN es crear un simple y comprensible mecanismo para la creación de modelos de procesos de negocio, y al mismo tiempo ser capaz de manejar la complejidad inherente a los procesos de negocio. El enfoque adoptado para manejar estos dos requisitos conflictivos fue la organización de la aspectos gráficos de la notación en categorías específicas. Esto proporciona un pequeño conjunto de categorías de notación para que el lector de un diagrama BPMN puede reconocer fácilmente los tipos básicos de elementos y entender el diagrama. Las cinco categorías básicas de elementos son:

1. **Objetos de flujo.** Son los principales elementos gráficos para definir el comportamiento de un proceso de negocio.
   a) **Eventos.** Un evento es algo que sucede durante el proceso. Este evento afecta el flujo del proceso y usualmente tiene una causa («trigger») y un impacto (resultado). Se representan con un círculo. De acuerdo con el momento en que afectan al flujo, se dividen en tres tipos: inicio, intermedio y fin.
      1) **Eventos Iniciales.** Indican donde comenzará el proceso.
      2) **Eventos Intermedios.** Son eventos que ocurren después de que el proceso ha iniciado y antes de que éste termine. Estos eventos pueden afectar el flujo del proceso pero no iniciarlo o directamente terminarlo. Pueden ser colocados en el flujo normal del proceso o agregados al límite de una actividad.
         - **Eventos Intermedios (Flujo Normal).** Cuando son colocados dentro del flujo normal del proceso representa cosas que suceden durante las operaciones normales del proceso. Pueden representar la respuesta a el evento (ej. La recepción de un mensaje). Pueden representar la creación de el evento (ej. El envío de un mensaje).
         - **Eventos Intermedios (Agregado al borde de una Actividad).** Los eventos que son agregados al borde o límite de una actividad indican que la actividad debe ser interrumpida cuando se dispara el evento. Se pueden agregar tanto a tareas como a subprocesos. Son usados para el manejo de errores, manejo de excepciones y compensaciones.
      3) **Eventos Finales.** Indican donde finalizará el proceso. Indican donde termina el flujo de secuencia del proceso y por tanto no tienen ningún flujo de secuencia saliente. Existen

Figura 1.1.: Tipos de eventos.

diferentes "resultados" que indican las circunstancias específicas por las que termina el proceso.

En la figura 1.1 se muestra un pequeño resumen de los tipos de eventos vistos anteriormente.

b) **Actividades.** Una actividad es un término genérico para describir el trabajo que realiza una compañía. Una actividad puede ser atómica o compuesta. Se representa con un rectángulo redondeado. Pueden ser realizadas una vez o pueden tener repeticiones definidas internamente. Los tipos de actividades que son una parte de un diagrama de proceso de negocio son:

1) **Tarea.** Una tarea es una actividad atómica que está incluida dentro de un proceso. Se habla de tarea cuando el trabajo que representa en el proceso no puede desglosarse en un nivel mayor de detalle.

   Generalmente un usuario final y/o una aplicación son usados para realizar la tarea cuando es ejecutada. Hay tipos especializados de tareas:

   - **Tarea de Servicio.** Es una tarea que utiliza algún tipo de servicio, lo que podría ser un servicio web o una aplicación automatizada. Son todas aquellas tareas que realiza el

Figura 1.2.: Tipos de tareas.

sistema sin intervención humana, como lo puede ser: enviar un «email» o invocar «web service».

- **Tarea de Envío.** Es una tarea simple que está diseñado para enviar un mensaje a un participante externo (en relación con el proceso).
- **Tarea de Recepción.** Es una tarea sencilla que se ha diseñado para esperar a que llegue un mensaje de un participante externo (en relación con el proceso).
- **Tarea de Usuario.** Es una típica tarea "flujo de trabajo", donde un intérprete humano realiza la tarea con la ayuda de una aplicación de software y se programa a través de un gestor de listas de tareas de una cierta clase.
- **Tarea Manual.** Es la que se espera que sea realizada sin la ayuda de un motor de procesos de negocio o cualquier otra aplicación. Un ejemplo de esto podría ser un técnico de teléfono de la instalación de un teléfono en una ubicación del cliente.
- **Tarea De Reglas de Negocio.** Proporciona un mecanismo para el proceso de hacer aportaciones a un «Business Rules Engine» y para obtener el resultado de los cálculos que el «Business Rules Engine» podría proporcionar.
- **Tarea «Script».** Se ejecuta mediante un motor de procesos de negocio. El modelador o implementador define una secuencia de comandos en un idioma que el motor puede interpretar. Cuando la tarea está lista para empezar, el motor va a ejecutar el «script». Cuando se ha completado la secuencia de comandos, se completará.

Existen atributos que determinarán si la tarea puede repetirse o si sólo se puede realizar una vez. Estos atributos son indicados mediante una marca y una tarea puede tener una o dos de estas marcas.

Tareas de Compensación.

Son usadas para la cancelación de una actividad realizada anteriormente. Están fuera del flujo normal del proceso y están asociadas a actividades normales.

En la figura 1.2 se muestran las representaciones de algunos tipos de tareas vistos con anterioridad.

2) **Subproceso.** Es una actividad cuyos detalles internos han sido modelados utilizando actividades, puertas de enlace, eventos y flujos de secuencia. Un subproceso permite el desarrollo jerárquico del proceso. Es una actividad compuesta que está incluida dentro de un proceso, que puede ser "abierto" para mostrar otro proceso de menor nivel. Subprocesos definen un ámbito contextual que se puede utilizar para la visibilidad de atributos, ámbito transaccional, para el manejo de excepciones, de eventos, o para la compensación. Puede desglosarse en diferentes niveles de detalle denominadas tareas. La versión colapsada de un subproceso, los detalles del mismo no son visibles en el diagrama. El signo más (+) indica que la actividad es un subproceso y que tiene un nivel más bajo de detalle. Esta asociado a un solo rol. En la versión expandida, los detalles de un proceso son visibles, es decir, esta en el mismo nivel de detalle del proceso y tiene un evento de inicio y fin del proceso. Puede estar asociado a uno o varios roles. Se representa con un símbolo de suma en la parte central inferior de la figura.

c) **«Gateways» (Puerta de enlace).** Se representa con un diamante, y son usados para controlar tanto la interacción como la convergencia y divergencia del flujo de secuencia dentro de un proceso. Éstas determinan ramificaciones, bifurcaciones, combinaciones y fusiones del proceso. Un nodo representa un punto en el proceso en donde el flujo necesita ser controlado.

1) **Exclusivos.**

- Los nodos de decisión exclusivos son colocados dentro de un proceso de negocio en donde el flujo de secuencia debe tener dos o más trayectorias alternativas.
- Sólo una de las posibles trayectorias resultantes puede ser seguida cuando el proceso es llevado a cabo.
- También son usados para unir el flujo de secuencia.
- Hay dos tipos de mecanismos de decisión:

  Basada en Datos. Son los tipos de nodos más comúnmente usados. El nodo de decisión crea trayectorias alternativas basadas en condiciones definidas (ej. Expresiones de condición).

  Basada en Eventos. Este tipo de decisión representa un punto de bifurcación en el proceso, en donde, las alternativas están basadas en eventos que ocurren en ese punto del proceso, más que en condiciones. El evento elegido en la decisión determina la trayectoria elegida (ej. La recepción de mensajes alternativos).

2) **Inclusivos.**

- Son decisiones en donde hay más de un resultado posible.
- Las trayectorias alternativas están basadas en expresiones condicionales contenidas dentro del flujo de secuencia resultante.
- La evaluación verdadera de una condición no excluye la evaluación de las otras condiciones.
- Al ser independiente cada trayectoria, todas las combinaciones pueden ser seguidas o al menos una de ellas.

3) **Complejos.**

- Son decisiones en donde hay definiciones más avanzadas de comportamiento.
- Considera situaciones que no son fácilmente tomadas a través de otros tipos de decisiones.
- Pueden ser usadas para combinar un conjunto de decisiones simples vinculadas en una situación única más compacta.

4) **Paralelos.**
   - Son lugares en el proceso en donde se definen múltiples trayectorias paralelas.
   - Proporcionan un mecanismo para sincronizar o para crear flujos paralelos.
   - Sirven para especificar cuando se pueden seguir dos trayectorias paralelas, aunque en realidad no son requeridos para crear flujos paralelos.

2. **Datos.** Los tipos de datos son los siguientes:
   a) **«Data Objects».** Se muestran visualmente en un diagrama de proceso y deben estar dentro de procesos o subprocesos. Son una manera de reutilizar objetos de datos en el mismo diagrama. Pueden ser usados para definir información de entrada/salida de las actividades y no tienen ningún efecto directo en los flujos de secuencia o mensaje del proceso. Pueden tener un "estado" que muestra como puede ser cambiado o actualizado dentro del proceso.
   b) **Entradas de datos.** Es una declaración de que un determinado tipo de datos se puede utilizar como entrada del InputOutputSpecification. Puede haber múltiples entradas de datos asociados a un «InputOutputSpecification».
   c) **Salidas de datos.** Es una declaración de que un determinado tipo de datos se puede producir como salida del «InputOutputSpecification». Puede haber múltiples salidas de datos asociados a un «InputOutputSpecification».
   d) **Almacenes de datos.** Proporciona un mecanismo de actividades para recuperar o actualizar la información almacenada que persistirá más allá del alcance del proceso.

3. **Objetos de Conexión.** Conectan los objetos de flujo en un diagrama para crear la estructura esquemática básica de un proceso de negocio, y definen el orden de ejecución de las actividades. Hay tres formas de conectar objetos de flujo con otros o con otra información: flujos de secuencia, de mensaje y de asociación .
   a) **Flujos de Secuencia.**
      - Un flujo de secuencia es usado para mostrar el orden en que se llevarán a cabo las actividades en un proceso.
      - Cada flujo tiene una sola fuente y un solo objetivo.
      - La fuente y el objetivo deben ser cualquiera de los siguientes objetos de flujo: Eventos (Inicio, Intermedios y Finales), Actividades (Tareas y Subprocesos) y «Gateways» (puertas de enlace).
   b) **Flujos de mensajes.**
      - Un flujo de mensaje es usado para mostrar el flujo de mensajes entre dos entidades que están preparadas para enviarlos y recibirlos.
      - Puede conectar a dos entidades («pools»), ya sea mediante los «pools» mismos o mediante objetos de flujo dentro de los «pools».
      - No pueden conectar dos objetos dentro de un mismo «pool».
   c) **Asociaciones.**
      - Una asociación es usada para asociar información y artefactos con objetos de flujo (eventos, actividades, «gateways»)
      - Textos y objetos gráficos puede ser asociados al flujo del proceso y a los objetos de flujo.
      - Son usadas para mostrar la manera en que los datos representan una entrada o una salida de las actividades .

- También es usada para mostrar las actividades usadas como compensación.

4. **«Swimlanes».** Son un mecanismo empleado para organizar actividades en categorías separadas visualmente, con el fin de ilustrar diferentes capacidades funcionales o responsabilidades. Hay dos principales tipos:

   a) **«Pools».** Representan a los participantes en un Diagrama de Proceso de Negocio Interactivo (B2B). Un participante puede ser un rol de negocios (ej. Comprador o vendedor) o una entidad de negocios (ej. ONU). Un «pool» puede ser una "caja negra" o puede contener un proceso. La interacción entre «pools» es manejada mediante flujos de mensaje. Los flujos de secuencia no pueden cruzar los límites de un «pool», esto es, un proceso debe estar totalmente contenido dentro de un «pool».

   b) **«Lanes».** Es una subpartición dentro de un proceso, a veces dentro de un «pool», y se extenderá a todo lo largo del proceso, ya sea vertical u horizontalmente. Los «lanes» son usados para organizar y categorizar actividades. A menudo representan roles organizacionales (ej. Administrador, Recepcionista) pero pueden representar cualquier característica deseada del proceso (roles internos, sistemas, departamentos internos). Los flujos de secuencia pueden cruzar los límites de un «lane» (Carril).

5. **Artefactos.** Son modeladores con la capacidad de mostrar información adicional acerca de un proceso, y no están directamente relacionados al flujo de secuencia o al flujo de mensaje del proceso. Los tipos de artefactos son:

   a) **Grupos.** Es un conjunto de elementos gráficos que se encuentran dentro de la misma categoría. Los grupos son artefactos que proporcionan un mecanismo visual para agrupar elementos de un proceso de manera informal. El agrupamiento puede ser usado para la documentación o por propósitos de análisis. Los grupos también pueden ser usados para identificar actividades de una transacción distribuida que es mostrada a través de «pools». El agrupamiento no afecta la secuencia o flujo de mensajes.

   b) **Anotaciones de Texto.** Son un mecanismo del modelador para dar mayor información a los lectores de un diagrama BPMN. Facilitan al lector de un diagrama de proceso de negocio su entendimiento. Pueden ser conectadas a un objeto específico en el diagrama con una asociación sin afectar el flujo del proceso.

Reglas de conexión

Existen reglas de conexión para los flujos de secuencia y los flujos de mensaje los cuales pueden ser colocados en cualquier dirección en un objeto de datos, ya sea a la izquierda, la derecha, arriba o abajo. BPMN permite esta flexibilidad, sin embargo cuando un diagrama contiene flujos de secuencia y de mensaje, es recomendable emplear una manera práctica de conectar los objetos de flujo de manera que al lector del diagrama le resulte fácil y claro de seguir. Resulta más fácil de entender los diagramas si los flujos de mensaje se colocan en un ángulo de 90° en relación a los flujos de secuencia. En la figura 1.3 las reglas de conexión para los flujos de secuencia y en la figura 1.4 las reglas de conexión para los flujos de mensaje. Los «pools», «lanes», objetos de datos y anotaciones de texto no pueden tener un flujo de secuencia entrante o saliente. Los «lanes», «gateways», objetos de datos y anotaciones de texto no pueden tener un flujo de mensaje entrante o saliente.

En la figura 1.5 se muestra un pequeño resumen de los elementos básicos de la notación gráfica básica para el diseño del diagrama.

En la figura 1.6 se puede observar un ejemplo con esta notación, donde el empleado inicia el proceso rellenando un formulario llamado "Solicitud" y el proceso recibe dichos datos en el evento de inicio de mensaje. A continuación, el proceso envía mediante un evento intermedio de mensaje un formulario llamado

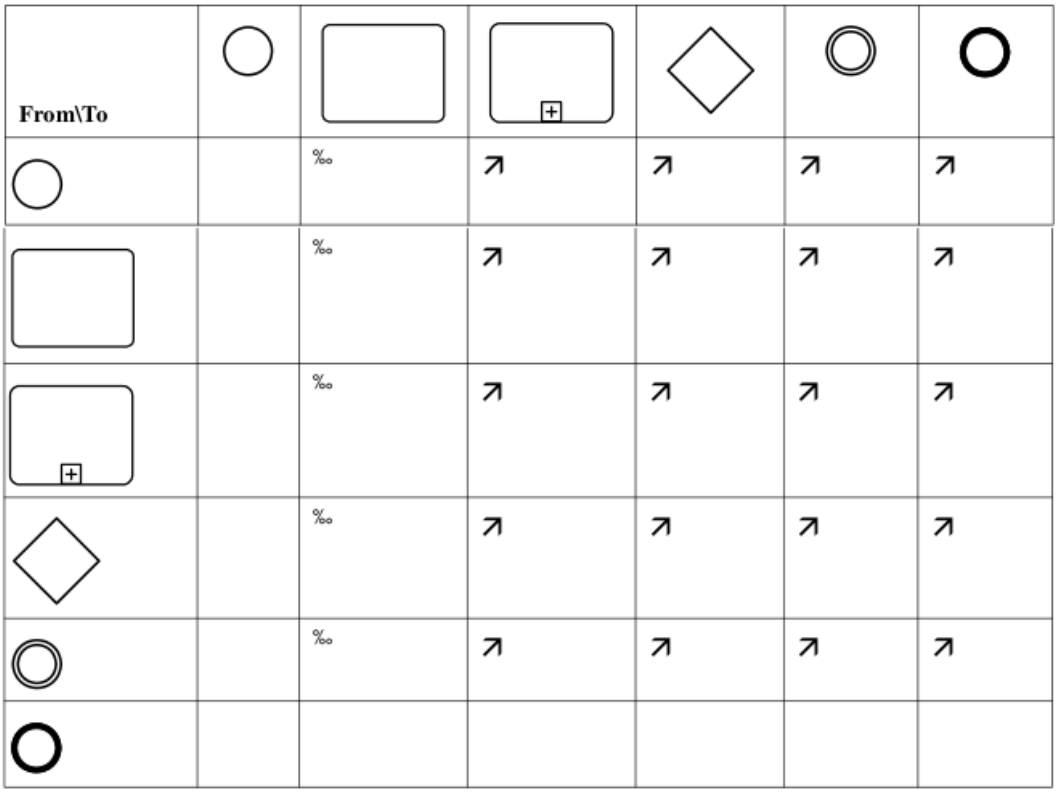


Figura 1.3.: Reglas de conexión para los flujos de secuencia.

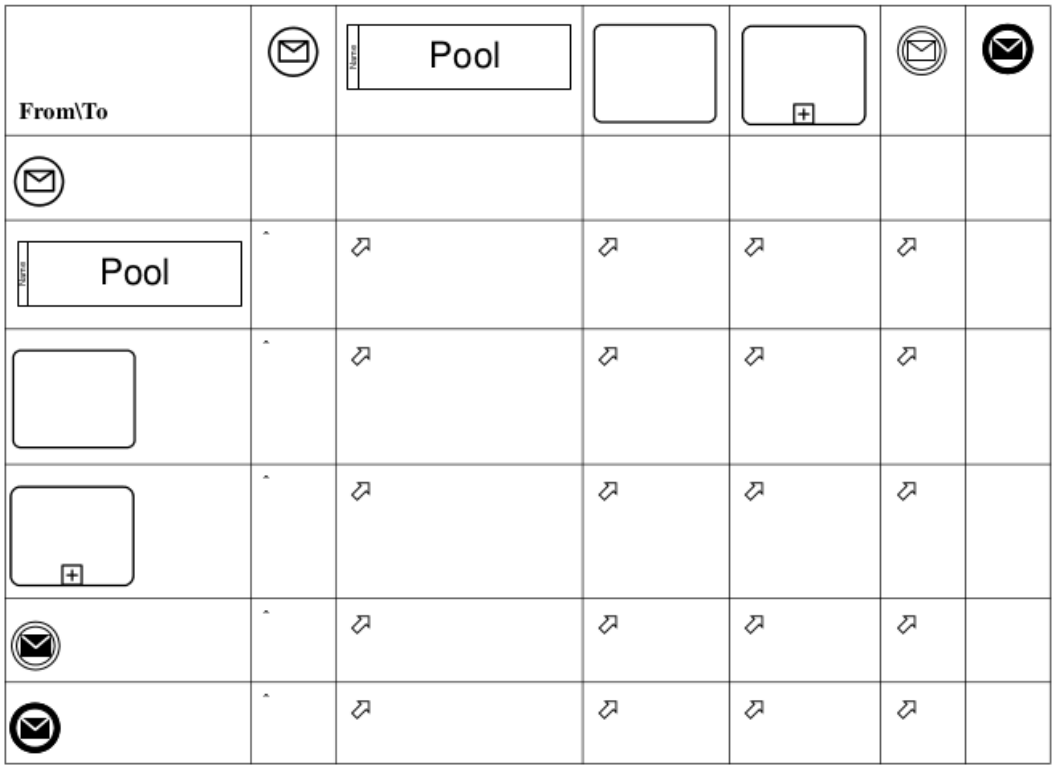


Figura 1.4.: Reglas de conexión para los flujos de mensaje.

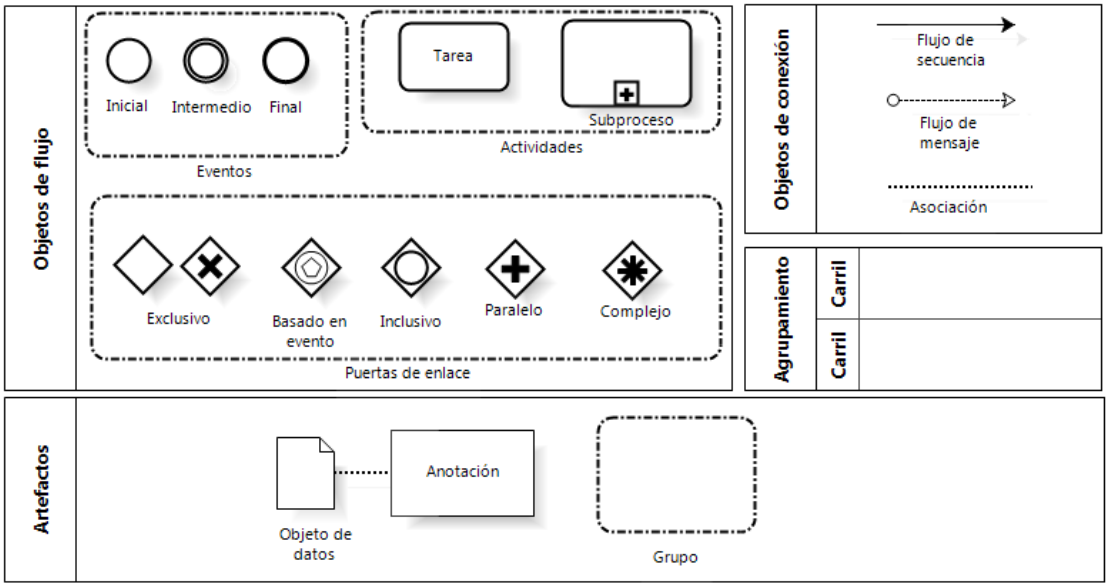


Figura 1.5.: Notación gráfica básica para el diseño del diagrama.

"ProcesoAGerente", donde, el gerente rellena los datos que faltan, entre ellos, confirmar si se aprueba o no la solicitud. El proceso recibe dichos datos en el evento intermedio de mensaje y seguidamente, se toma un camino u otro dependiendo del «gateway» exclusivo si se ha aprobado o no dicha solicitud. Si se aprueba, mediante un evento final de mensaje se envía una notificación a través de un formulario llamado "Mensajes". Si no se aprueba, se envía a través de un evento intermedio de mensaje otro formulario llamado "ProcesoAEmpleado" y que el empleado corrige los datos que no han sido aprobados. El proceso recibe dichas correcciones en un evento intermedio de mensaje y finalmente, mediante un evento final de mensaje se envía una notificación a través de un formulario llamado "Mensajes".

1.4.8. Maven

Maven [9, 10] es una herramienta de gestión de proyecto que utiliza un Project Object Model (POM), un conjunto de normas, un ciclo de vida del proyecto, un sistema de gestión de la dependencia, y la lógica de la ejecución de los objetivos del «plugin» en las fases definidas en un ciclo de vida. Cuando se utiliza Maven, se describe el proyecto con un POM bien definido, Maven puede entonces aplicar la lógica transversal de un conjunto de «plugins» compartidos (o personalizados). Es una herramienta diseñada para la gestión y construcción de proyectos Java. Fue creada por «Jason van Zyl», de Sonatype, en 2002.

Apache Maven se originó como un intento de simplificar el proceso de construcción para el proyecto ya desaparecido Apache Jakarta Alejandría. En sus inicios fue un proyecto interno en Apache Turbine donde esto tarde o temprano vino para sustituir un sistema de construcción quebradizo y frágil basado en Apache Ant.

Considerando la potencia enorme de Maven y la eficacia en la solución de una mayoría de nuestros desafíos cotidianos, se ha hecho enormemente popular y ahora es ampliamente usado no sólo por los desarrolladores, sino por otros roles en un equipo que incluye a maestros scrum, los propietarios del producto, y los jefes de proyecto . En los últimos años, Maven claramente ha surgido como aumento significativo de fortaleza para equipos ágiles y organizaciones.

Cada proyecto Maven tiene lo que se conoce como un objeto del modelo de proyecto (POM) en un archivo llamado pom.xml. Este archivo describe el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos

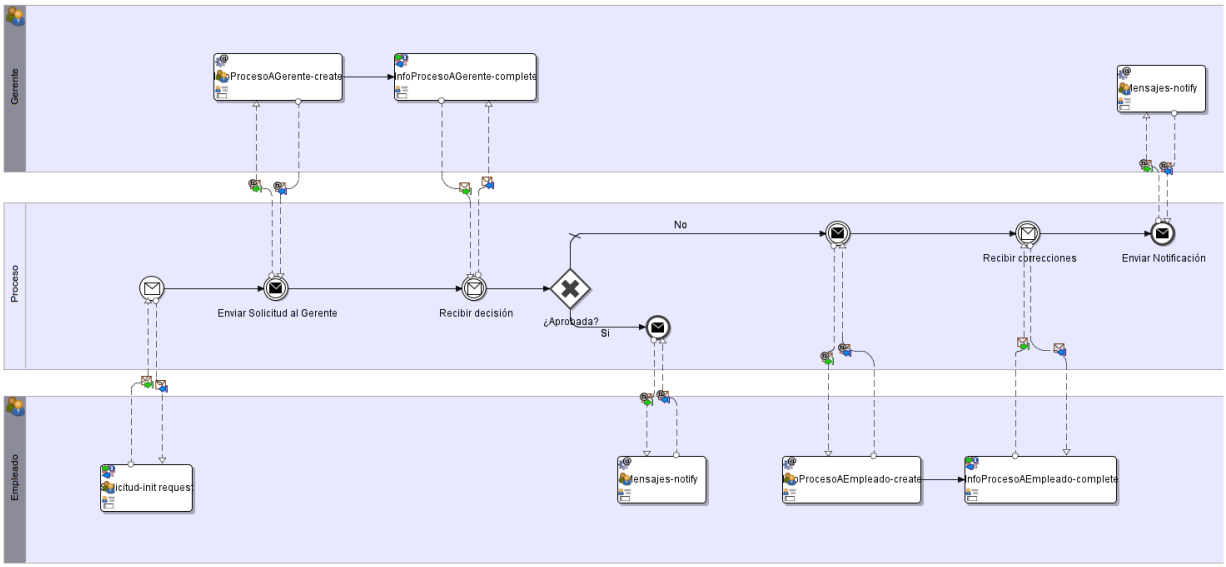


Figura 1.6.: Ejemplo de un diagrama con notación BPMN de una solicitud de un empleado a un gerente.

predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Las partes importantes sería la definición de `groupId` que representa la organización propietaria del artefacto, cumple una función similar a la de un paquete Java. `artifactId` sería el nombre de este artefacto (jar) en el repositorio. Como es de suponer `version` define el número de versión del artefacto. Y bajo `dependencies` aparecen las dependencias, que como se puede ver constan precisamente de un `groupId`, un `artifactId` y una `version`.

Otro concepto importante dentro de Maven es el de repositorios. Un repositorio es un almacén de paquetes (archivos jar habitualmente) ordenados de forma inequívoca incluyendo la versión. Maven se descarga a un repositorio local todas las dependencias y «plugins» que va necesitando. Dicho repositorio provee acceso a muchas versiones de diferentes proyectos «Open Source» en Java, de Apache y de otras organizaciones. Este repositorio y su sucesor reorganizado, Maven 2, intentan ser el mecanismo por defecto de distribución de aplicaciones en Java, pero su adopción está siendo lenta. Maven permite subir artefactos al repositorio al final de la construcción de la aplicación, de forma que cualquier usuario tiene acceso a ella.

Usa una arquitectura basada en «plugins» que permite que utilice cualquier aplicación controlable a través de la entrada estándar. En teoría, esto permite que cualquiera pueda escribir sus propios «plugins» para su interfaz con herramientas como compiladores, herramientas de pruebas unitarias, etc. para cualquier otro lenguaje; pero en la realidad, Maven apenas soporta otros lenguajes distintos a Java. Está construido alrededor de la idea de reutilización de la lógica de construcción: los proyectos se construyen generalmente con patrones similares; una elección lógica sería reutilizar los procesos de construcción. La idea no es reutilizar el código o funcionalidad, sino cambiar la configuración o también código escrito.

Los proyectos en Maven cuentan con una serie de etapas, llamadas ciclo de vida. Para pasar de etapa, es necesario haber completado con éxito las etapas anteriores. Estas etapas representan las distintas fases por las que un proyecto software ha de pasar. Hay tres ciclos de vida incorporado construcción: «default», «clean», y «site». El ciclo de vida por defecto controla la compilación del proyecto, prueba y despliegue. A pesar de que contiene más de 20 fases de construcción, las siguientes son las fases más importantes:

- **«Validate»:** Valida que toda la información del proyecto está disponible y es correcta.
- **«Compile»:** Compila el código fuente.

- **«Test»:** Pruebas unitarias se ejecutan dentro de un marco adecuado.
- **«Package»:** Paquetes del código compilado en su formato de distribución.
- **«Integration-test»:** Procesa el paquete en el entorno de integración-test.
- **«Verify»:** Ejecuta comprobaciones para verificar que el paquete es válido.
- **«Install»:** Instala el paquete en el repositorio local.
- **«Deploy»:** Instala el paquete final en un repositorio remoto.

El ciclo de vida «clean» se encarga de la limpieza del proyecto y contiene las siguientes fases de construcción:

- **«Pre-clean»:** Ejecuta los procesos necesarios antes de la limpieza del proyecto.
- **«Clean»:** Elimina todos los archivos generados.
- **«Post-clean»:** Ejecuta los procesos necesarios para completar la limpieza del proyecto.

El ciclo de vida «site» se encarga de la generación y distribución de la documentación del sitio del proyecto:

- **«Pre-site»:** Ejecuta los procesos necesarios antes de la generación del sitio.
- **«Site»:** Genera la documentación del sitio del proyecto.
- **«Post-site»:** Ejecuta los procesos necesarios para completar la generación de sitio y se prepara el sitio para el despliegue.
- **«Site-deploy»:** Despliega la documentación sitio en el servidor web especificado.

1.4.9. BPELUnit

BPELUnit [4] es un marco de pruebas unitarias automáticas, repetibles y de caja blanca para probar composiciones WS-BPEL. El marco BPELUnit permite realizar pruebas de procesos BPEL mediante la especificación de un caso de prueba para el proceso BPEL. El caso de prueba consta de secuencias de operaciones de entrada/salida que son ejecutadas por el marco BPELUnit en nombre del cliente y de todos los socios del proceso BPEL, forzando así el proceso BPEL para ejecutar un determinado camino. Un caso de prueba pasa si todas las operaciones se han llevado a cabo con éxito. Puede usar cualquier motor que implemente WS-BPEL 2.0 e integrarse con Apache Ant y Eclipse.

Estos casos de prueba se definen en un fichero con extensión BPTS. Son archivos que especifican los casos de prueba para BPELUnit, están basados en XML y pueden ser creados y editados en cualquier editor de texto. Se puede especificar qué datos serán enviados y qué datos esperan ser recibidos por el cliente (`clientTrack`) y cada socio del proceso (`partnerTrack`), no siendo necesario que todos los socios interactúen en un determinado caso de prueba. BPELUnit actuará como cliente ejecutando los casos de prueba especificados en ellos o bien si se realiza una llamada a un servicio externo, BPELUnit sustituye el servicio externo con otro servicio («mockup») que se comporta del modo indicado en el caso de prueba.

El elemento raíz de un fichero BPTS se denomina `testsuite` y consta los siguientes elementos:

- **`name`:** especifica el nombre identificativo del conjunto de pruebas.

- **baseURL:** BPELUnit utiliza una Uniform Resource Locator (URL) local para simular los socios del proceso. La URL de cada socio es una concatenación de la URL base y el nombre del socio. La URL base por defecto es `http://localhost:7777/ws`, que se puede cambiar a cualquier dirección (local).

Está dividido en dos partes:

- **deployment.** Contiene la información relativa al despliegue del proceso y consta de:

  put: Detalla el proceso a desplegar mediante, al menos con, el nombre del proceso, el motor de ejecución y el fichero WSDL con la especificación del servicio.

  partner: Se especifican los socios del proceso, que pueden ser varios o ninguno. Cada socio se especifica mediante el identificador del «partner» y mediante el fichero WSDL con la especificación del servicio.

- **testCases.** Contiene un número arbitrario de casos de prueba, cada caso de prueba (`testCase`) se compone de una serie de atributos, se identifica mediante un nombre. Además, con el atributo `basedOn` un caso de prueba puede estar basado en uno anterior, con el atributo `abstract` indica a BPELUnit si puede ejecutar el caso de prueba («false») o no («true»), al ser un caso de prueba abstracto y con el atributo `vary` especifica si el caso de prueba se debe ejecutar varias veces. Cada caso de prueba a su vez contiene un único `clientTrack` y varios o ningún `partnerTracK` con actividades, que pueden ser las siguientes:

  sendOnly: Envía un único mensaje.

  receiveOnly: Espera un único mensaje y lo verifica.

  sendReceive: Se envía un único mensaje, espera una respuesta síncrona y la verifica.

  receiveSend: Espera un único mensaje, lo verifica y envía una respuesta síncrona.

  sendReceiveAsynchronous: Se envía un único mensaje, se espera una respuesta asíncrona y se verifica.

  receiveSendAsynchronous: Se espera un único mensaje, lo verifica y envía una respuesta asíncrona.

BPELUnit hace especial énfasis en los siguientes puntos:

- Apoyo a artículos BPEL específicos como manejo de actividad paralela, compensación y mensajería asíncrona.
- Facilidad de uso para el desarrollador.
- Una visión realista de la ejecución de un proceso BPEL.

El grupo UCASE utiliza este marco de pruebas unitarias. Un documento con extensión BPTS tiene una estructura semejante a la siguiente:

```
1 <testsuite>
2   <deployment>
3     <put name=``...''>
4
5     </put>
6
7     <partner name=``...''>
8
9     </partner>
```

```
10
11      <partner name=''...''>
12
13      </partner>
14    <deployment>
15
16    <testCases>
17      <testCase name=''...''>
18
19      </testCase>
20
21      <testCase name=''...''>
22
23      </testCase>
24    </testCases>
25
26  </testsuite>
```

2. Planificación

Este capítulo va a explicar de forma detallada las distintas etapas en las que se ha llevado a cabo el desarrollo de este PFC, además, se mostrará la cronología, mediante un diagrama de Gantt y se detallará también, la metodología que ha sido utilizada durante su desarrollo.

La realización de este PFC se ha llevado a cabo durante un periodo aproximado de 25 meses, concretamente desde mayo de 2013 hasta junio de 2015. Tras varios meses asistiendo a seminarios del grupo de investigación UCASE, en mayo, se obtuvo la documentación de este proyecto. Ha requerido un gran esfuerzo de planificación, debido a que no se ha tenido una dedicación exclusiva a su elaboración. Durante los primeros meses, el proyecto se compaginó con los estudios del primer ciclo de Ingeniería Informática y con seminarios semanales formativos o magistrales con el grupo UCASE, en los que se impartían cursos sobre herramientas útiles y afines al espíritu del grupo, o sesiones en las que se exponían los avances o dificultades encontradas en cada uno de los proyectos que se llevan a cabo. En los meses siguientes, se compatibilizó con unas prácticas de empresa en Cádiz, realizadas por las mañanas desde agosto de 2013, hasta agosto de 2014.

2.1. Metodología

Para desarrollar este proyecto se ha utilizado el modelo de construcción de prototipos. Esta metodología consiste en la creación de un prototipo que va a ayudar a definir los requisitos del sistema, es un elemento de trabajo. El prototipo debe ser construido en poco tiempo, usando los programas adecuados y herramientas de generación rápida de software. El prototipo es evaluado y se van refinando los requisitos del software que se desarrollará. Este modelo se ha aplicado de forma iterativa hasta conseguir que la aplicación cumpliera con los requisitos esperados. El prototipo es una primera aproximación al sistema y lo recomendable es desecharlo una vez que haya cumplido su función y reconstruir el software con una visión hacia la calidad y la facilidad de mantenimiento. Se suele aplicar este modelo cuando no se reconocen muy bien todos los requisitos del sistema al comienzo del desarrollo, los requisitos evolucionan muy rápidamente y hay dudas sobre alguna parte del sistema.

2.2. Fases del proyecto

Ahora se van a definir todas y cada una de las fases por las que ha sido necesario pasar hasta finalizar la aplicación:

2.2.1. Fase 0: Recopilación de la información

En esta fase, se comenzó la toma de contacto con el grupo de investigación UCASE, asistiendo a los seminarios una vez a la semana durante aproximadamente 6 meses para obtener una formación previa y comprender nuevos conceptos. El grupo trabaja con WS-BPEL o BPEL que es un lenguaje de programación que describe procesos de negocios que orquestan servicios web. Tras esta época de preparación, se comenzó a realizar este proyecto. Consiste en desarrollar un caso de estudio realista en el que se invoque a servicios web reales con su propia lógica de negocio y capa de persistencia, y en el que los usuarios realicen algunas tareas de forma manual. Son composiciones WS-BPEL para la gestión inmobiliaria como se ha comentado en el capítulo anterior.

Se obtuvo la documentación del proyecto, como se ha llevado acabo en Suiza, se encontraba escrita en francés y algunos títulos o imágenes en alemán ya que Suiza tiene cuatro idiomas oficiales: el alemán, el francés, el italiano y el romanche, y por lo tanto, se tradujo toda la documentación al español. Una vez traducida la documentación ya se podía comenzar a analizar los requisitos necesarios para este PFC. Fue necesario tener reuniones con los tutores del proyecto, para analizar dichos requisitos.

2.2.2. Fase 1: Análisis y diseño del proyecto

Después de haberse leído la documentación y tener la definición de los requisitos funcionales, se prosiguió realizando el análisis y el diseño del proyecto. Esta fase tiene como objetivo, plantear cómo se van a implementar todas las funcionalidades del sistema, así como las estructuras y tipos de datos que se van a utilizar. Por último, destacar que en esta fase se ha tenido que investigar sobre la gestión inmobiliaria en Suiza para el diseño la base de datos y las validaciones de cada uno de los datos.

2.2.3. Fase 2: Aprendizaje de las tecnologías a utilizar

Una vez obtenidos los requisitos del proyecto se pasó a estudiar las tecnologías, herramientas y lenguajes que se han utilizado para la realización de este proyecto. En concreto, el lenguaje WS-BPEL 2.0, se usa para la realización de las composiciones, facilitada la especificación por el tutor. Además, se ha estudiado otras composiciones para tenerlas de referencia. En esta etapa también, se han aprendido otras tecnologías como WSDL, BPELUnit, XML Schema, Xpath 2.0, XQuery 1.0 y SOAP. También, para la creación de dichas composiciones, se ha estudiado el uso de la herramienta Intalio|Business Process Management System (BPMS) [24] y la tecnología BPMN, además de la parte teórica, hicieron falta crear algunos pequeños programas de ejemplo en Intalio para poder comenzar con las composiciones que eran más complejas.

Además, consistió en aprender el lenguaje de programación Java, no se había trabajado anteriormente con él. Gracias a su similitud al lenguaje C++, no fue especialmente dificultoso. Además de maven, para la creación de un proyecto multimódulo, DbUnit y JUnit [25] para las pruebas unitarias. Entre las herramientas, ya se había trabajado con Eclipse y Tomcat, he aprendido a utilizar el control de versiones (subversive), MySQL [26] para la base de datos, Sonar [27] y Jenkins [28], para mejorar la calidad del software, Kile [29] junto con el lenguaje LaTeX [30], para el desarrollo de esta memoria, y otros como, Dia [31], ArgoUml [32] y GanttProject [33] para el desarrollo de los diagramas.

Para decidir las herramientas concretas se tuvo en cuenta las propuestas y consejos de mis tutores. Fue necesario un gran esfuerzo, ya que la mayoría de las tecnologías y lenguajes utilizados no se habían usado con anterioridad.

2.2.4. Fase 3: Elección de herramienta gráfica para el desarrollo de composiciones

En esta fase, se realizó un estudio para elegir que herramienta gráfica utilizar para el desarrollo de las composiciones WS-BPEL 2.0. Mis tutores me propusieron la herramienta Intalio|BPMS y finalmente, se decide utilizarla por las características que presenta y su facilidad de uso, es una herramienta BPEL de código abierto, basada en Apache ODE [34], Axis2 y Apache Geronimo, que permite diseñar modelos en la notación BPMN. Está compuesta por: Intalio|Designer: herramienta utilizada para diseñar los modelos e Intalio|Server: servidor BPEL que ejecuta los procesos. A partir del modelo BPMN la herramienta genera el código BPEL a ejecutar en Intalio|Server, y define los archivos WSDL y XSD para que los servicios web puedan ser invocados por un cliente. Los clientes y proveedores se comunican con el protocolo de mensajes SOAP, escrito en XML y utilizado para enviar requerimientos y recibir respuestas. Más características:

- Fácil de Usar.

- Su modelo de objetos de tareas es extensible.
- Proporciona tareas para aceptar, completar, cancelar, reasignar, etc.
- Cuenta con un marco de seguridad basado en Role Based Access Control (RBAC) y Single Sign-On (SSO).
- Cuenta con un set de procesos BPEL definidos para el «workflow».
- Cuenta con servicios para el despliegue de las tareas, etc.
- Soporte de «Attachments».
- Interfaces basadas en «web service» y Representational State Transfer (REST).

2.2.5. Fase 4: Desarrollo de la capa de persistencia, de dominio y los servicios web

En esta fase, se implementan la capa de persistencia y de dominio del sistema y los distintos servicios web básicos a los que llaman las composiciones WS-BPEL. Se comenzó creando un proyecto maven padre y luego los módulos, uno para la capa de persistencia y de dominio y otro para los servicios web necesarios que llaman las composiciones WS-BPEL. Seguidamente, se implementó la capa de persistencia y de dominio con la ayuda de un libro de referencia, ya que, no se tenía experiencia en ello. Finalizada la capa de persistencia y de dominio, se crearon los servicios web de las típicas Create Read Update Delete (CRUD) y otros más específicos necesario para el desarrollo de las composiciones.

2.2.6. Fase 5: Desarrollo de las composiciones WS-BPEL

Se comenzó instalando la herramienta Intalio, elegida como se ha comentado anteriormente en otra fase, y su servidor. Se realizaron diferentes pruebas más pequeñas para ir familiarizándose con el programa. Tras tener práctica suficiente, ya se podía seguir realizando esta etapa.

En esta fase, se desarrollan las composiciones WS-BPEL. Los pasos a seguir para el desarrollo para cada composición van a ser los mismos. Se realiza el modelado de cada composición en BPMN y una vez correcto y finalizado, se prueban en el servidor y se puede descargar la composición entera en un archivo que genera el programa. Para cada composición se desarrolla un fichero BPEL en el que se define la lógica del proceso, un fichero WSDL en el que se definen todas las características necesarias para los servicios que se utilizan en la composición.

2.2.7. Fase 7: Pruebas

En esta etapa se realizan las pruebas, son algo fundamental en este proyecto para producir un software de la mejor calidad posible, que cumpla, y si puede, supere las expectativas. Por lo tanto, es una etapa muy importante. Para ello se han utilizado DbUnit y JUnit, para las pruebas con Java, se comprobó que la capa de persistencia y de dominio funcionaban correctamente junto con la base de datos de MySQL, y BPELUnit para las pruebas de las composiciones. Las pruebas de los servicios web se hicieron de forma manual.

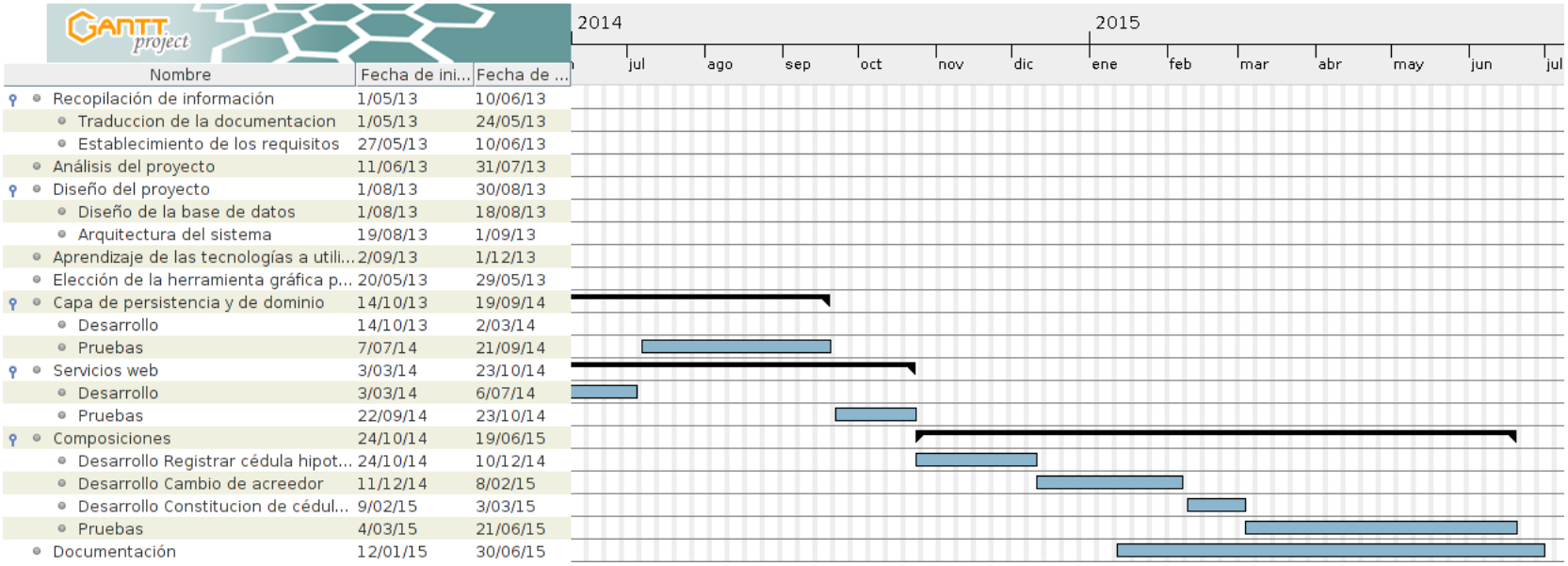
2.2.8. Fase 8: Documentación

En esta etapa, se ha realizado la memoria para este PFC. Se ha ido desarrollando desde el comienzo del estudio, aunque los últimos meses se ha ido intensificando su desarrollo. Para la realización de la misma he usado el lenguaje LaTeX, que facilita el desarrollo de documentos de grandes dimensiones.

2.3. Distribución temporal

Para representar gráficamente las diferentes fases, tareas y actividades programadas como parte del proyecto y para mostrar la distribución temporal que se le ha dedicado a las distintas fases de este PFC, se ha empleado la herramienta Gantt Project para elaborar un diagrama de Gantt. En las figuras 2.1 y 2.2 se puede observar.

Figura 2.1.: Diagrama de Gantt. Primera Parte.



Nombre	Fecha de ini...	Fecha de ...
Recopilación de información	1/05/13	10/06/13
Traduccion de la documentacion	1/05/13	24/05/13
Establecimiento de los requisitos	27/05/13	10/06/13
Análisis del proyecto	11/06/13	31/07/13
Diseño del proyecto	1/08/13	30/08/13
Diseño de la base de datos	1/08/13	18/08/13
Arquitectura del sistema	19/08/13	1/09/13
Aprendizaje de las tecnologías a utili...	2/09/13	1/12/13
Elección de la herramienta gráfica p...	20/05/13	29/05/13
Capa de persistencia y de dominio	14/10/13	19/09/14
Desarrollo	14/10/13	2/03/14
Pruebas	7/07/14	21/09/14
Servicios web	3/03/14	23/10/14
Desarrollo	3/03/14	6/07/14
Pruebas	22/09/14	23/10/14
Composiciones	24/10/14	19/06/15
Desarrollo Registrar cédula hipot...	24/10/14	10/12/14
Desarrollo Cambio de acreedor	11/12/14	8/02/15
Desarrollo Constitucion de cédul...	9/02/15	3/03/15
Pruebas	4/03/15	21/06/15
Documentación	12/01/15	30/06/15

Figura 2.2.: Diagrama de Gantt. Segunda Parte.

3. Análisis del proyecto

Este capítulo se basa en conseguir la especificación detallada del sistema, se va a realizar un estudio de los requisitos que este debe cumplir; a partir de estos se realizarán unos procesos para definir su estructura conceptual. Dichos procesos que se van a llevar a cabo para la realización del análisis son el modelo de casos de uso, conceptual de datos y comportamiento del sistema.

3.1. Introducción

Las letras hipotecarias son uno de los derechos que están registrados y sólo son expedidas por el registro de la propiedad. La banca suiza se caracteriza por su estabilidad, privacidad y protección de los activos e información de sus clientes. Además, el valor del franco suizo (CHF) se ha mantenido relativamente estable comparado con el de otras monedas. Los bancos suizos, así como el correo suizo, utilizan un sistema de pagos electrónicos conocido como Swiss Interbank Clearing. Los créditos hipotecarios son uno de los principales pilares de la industria bancaria suiza. Durante décadas, los propietarios de viviendas y la economía se han beneficiado de bajas tasas de interés y de aumento constante de los precios de origen al solicitar créditos hipotecarios garantizados por letras hipotecarias. Aún así, los procesos de comunicación y de negocios nunca fueron estandarizados con éxito, ni digitalizados, a lo largo de toda Suiza entre bancos, notarios, y registros de la propiedad.

Hay necesidad de digitalizar ciertos procesos, y así, las oficinas del registro de la propiedad, los notarios y las entidades de crédito realicen sus operaciones de forma más eficiente y el trabajo administrativo disminuye. Por ello, la existencia de dicho caso de estudio basado en la documentación de un proyecto de gobierno electrónico suizo. Como se comentó en el primer capítulo se ha simplificado para adaptarlo a este proyecto y en base a esto, los procesos que se han desarrollado en este PFC son:

- Registrar cédula hipotecaria. Las cédulas hipotecarias podrán ser registradas.
- Cambio de acreedor / Recompra de créditos entre entidades de crédito. Cesión de los derechos del acreedor sobre una cédula hipotecaria.
- Constitución de cédulas hipotecarias de registro. La constitución de la cédula se hará bien mediante un contrato de garantía.

Las cédulas hipotecarias son valores emitidos por particulares, mediante la intervención de un banco hipotecario y garantizadas con hipoteca sobre un inmueble propiedad del emisor, además de la garantía solidaria del banco hipotecario que interviene en su emisión.

3.2. Requisitos del sistema

Los requisitos [35] son las capacidades y condiciones con las cuales debe ser conforme el sistema. El primer reto del trabajo de los requisitos es encontrar, comunicar y recordar lo que se necesita realmente, de manera que tenga un significado claro para el cliente y los miembros del equipo de desarrollo.

3.2.1. Requisitos funcionales

Los requisitos funcionales definen qué debe hacer un sistema. Los requisitos funcionales son:

- Permitir registrar la cédula hipotecaria existente en papel.
- Permitir cambiar el acreedor de la hipoteca en papel.
- Permitir cambiar el acreedor de la hipoteca ya registrada.
- Permitir constituir la cédula hipotecaria de registro.
- Permitir almacenar documentos de forma local.

3.2.2. Requisitos no funcionales

Los requisitos no funcionales definen cómo debe ser el sistema. Los requisitos no funcionales son:

- Sistema cerrado que acepta sólo clientes autorizados, siendo las oficinas del registro de la propiedad, funcionarios públicos (notarios), y entidades de crédito (bancos, seguros y fondo de pensiones).
- Sistema no accesible a los propietarios inmobiliarios privados. Los ciudadanos seguirán visitando a la entidad de crédito para las solicitudes del crédito y a un notario para los asuntos relativos al registro de la propiedad.
- Dependiendo del usuario o rol que tenga se puede comenzar diferentes procesos, se puede realizar distintas tareas y se puede recibir distintas notificaciones.

3.3. Modelo de análisis en UML

Tras finalizar la etapa anterior y conocer todos los requisitos del sistema, se ha analizado el comportamiento del sistema para llevarlos a cabo. Se ha usado el Lenguaje Unificado de Modelado (UML) [36] es un lenguaje para visualizar, especificar, construir y documentar los modelos de un sistema desde una perspectiva orientada a objetos. El UML es sólo un lenguaje, por lo que es sólo una parte de un método de desarrollo de software. El UML es un proceso independiente, aunque de manera óptima que se debe utilizar en un proceso que es impulsado de casos de uso, centrado en la arquitectura, iterativo, e incremental. Un lenguaje de modelado es un lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema. Un lenguaje de modelado como UML es, pues, un lenguaje estándar para los modelos de software.

El UML proporciona una forma estándar de escribir los modelos de un sistema, que cubre las cosas conceptuales tales como procesos de negocio y funciones del sistema, así como las cosas concretas, como las clases escritas en un lenguaje específico de programación, esquemas de bases de datos y componentes de software reutilizables.

Los procesos que se han realizado son los siguientes:

1. Modelo de casos de uso.
2. Modelo conceptual de datos.
3. Modelo comportamiento del sistema.

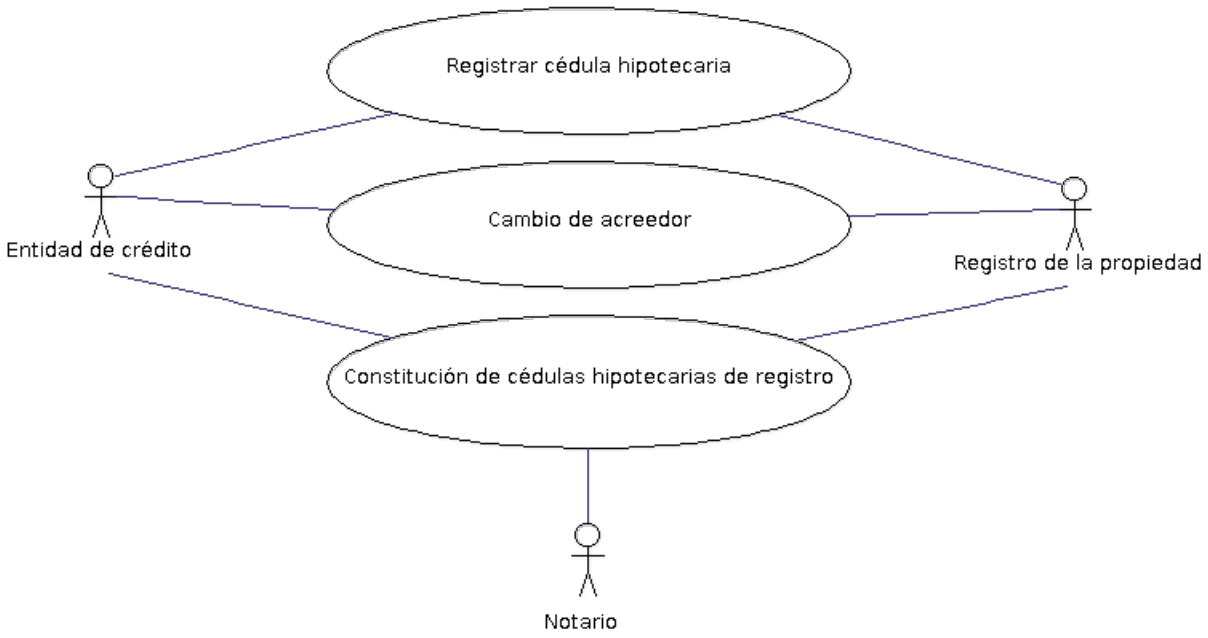


Figura 3.1.: Diagrama de Casos de Usos generales.

3.4. Modelado de casos de uso

Primero de todo, se ha aplicado el Modelo de Casos de Usos, describe la funcionalidad propuesta del nuevo sistema. Para poder definir el caso de uso se debe conocer antes varias definiciones. Un actor es algo con comportamiento, como una persona (identificada por un rol), sistema informatizado u organización; por ejemplo un cajero. Un escenario es una secuencia específica de acciones e interacciones entre los actores y el sistema objeto de estudio.

A continuación, si se puede definir que un caso de uso [35] es una colección de escenarios con éxito y fallo relacionados, que describe a los actores utilizando un sistema para satisfacer un objetivo. Los casos de uso representan los requisitos funcionales que indican que hará el sistema.

3.4.1. Diagrama de casos de uso

Se pasa a mostrar el diagrama de casos de uso que van a representar la funcionalidad del sistema. Los diagramas de casos de uso son importantes para modelar el comportamiento de un sistema, un subsistema o una clase. Cada uno muestra un conjunto de casos de uso, actores y sus relaciones. Es una notación gráfica que sirve para representar los diferentes casos de uso que conforman el modelo de casos de uso de forma simple y general.

En la figura 3.1 se puede ver el diagrama de casos de uso. Como se puede observar, se tiene 3 actores y 3 casos de uso. Los diagramas de casos de usos son una manera de verlos representados, lo fundamental en el modelado de casos de uso es la especificación de éstos.

3.4.2. Especificación de casos de uso

Descripción detallada de todos y cada uno de los casos de uso anteriormente representados en el diagrama.

Caso de uso: Registrar cédula hipotecaria

Identificación de los escenarios:

- **Escenario Principal**: Registro cédula hipotecaria OK (Declaración de acuerdo del propietario establecida previamente).
- **Escenario Alternativo**: Registro cédula hipotecaria OK (Declaración de acuerdo del propietario establecido directamente).
- **Escenario de error 1**: El inmueble ya existe.
- **Escenario de error 2**: El inmueble no está en el mismo distrito del registro de la propiedad.
- **Escenario de error 3**: La hipoteca ya existe.
- **Escenario de error 4**: Los datos a registrar no son correctos y la transacción ya existe.
- **Escenario de excepción**: Cancelar transacción.

Actores: Entidad de crédito (Actor principal) y oficina del registro de la propiedad (Actor secundario).

Descripción: Se realiza el registro de una cédula hipotecaria en papel determinada.

Precondiciones:

1. La oficina del registro de la propiedad afectada está conectada al sistema.
2. La entidad de crédito es participante del sistema, siendo bancos, los seguros y las cajas de pensiones.

Postcondiciones: El sistema registra la cédula hipotecaria en papel.

Escenario Principal:

1. El usuario de la entidad de crédito inicia el proceso e introduce la variante del acuerdo.
2. El sistema recibe mediante un mensaje la variante del acuerdo introducida por el usuario.
3. El sistema recibe mediante un mensaje la declaración de acuerdo del propietario por medio del formulario “Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro” completado por el usuario.
4. El sistema recibe un mensaje del usuario con el EGRID o número del inmueble y comprueba que el inmueble no existe en la base de datos y envía un mensaje con la confirmación.

   Se repite el paso 4 hasta que no haya más inmuebles.
5. El usuario de la entidad de crédito añade los datos que faltan de la orden enviada por el sistema.
6. El sistema recibe mediante un mensaje la orden que le envía el usuario de la entidad de crédito y comprueba que el inmueble si está en el mismo distrito que la oficina del registro de la propiedad.

   Se repite el paso 6 hasta que no haya más inmuebles.
7. El sistema añade el número de transacción enviándolo a través de un mensaje el formulario (envío de notificación) a la entidad de crédito.
8. El usuario de la entidad de crédito envía el formulario completado al sistema que lo recibe mediante un mensaje con los datos.

9. El sistema envía los documentos mediante un mensaje al registro de la propiedad (Declaración de acuerdo (Requisa), envío de notificación y la orden).

10. El usuario de la oficina del registro de la propiedad recibe los documentos rellenados mediante un mensaje (cédula (orden), declaración (requisa) y envío de notificación) y añade los datos que faltan de la orden enviada por el sistema (EREID).

11. El sistema recibe mediante un mensaje la orden que le envía el usuario de la oficina del registro de la propiedad y aprueba que es correcta (Hipoteca no existe en la Base de Datos (BD)),

    1.- El sistema registra los datos de la orden en la base de datos.

12. El sistema aprueba que los datos a registrar son correctos y la inscripción de la transacción en la BD (la transacción no existe en la BD):

    1.- El sistema registra la transacción en la base de datos.

    2.- La cédula hipotecaria en papel será anulada y destruida.

    3.- La cédula hipotecaria en papel salió del control de títulos.

    4.- Se registró la cédula hipotecaria en el registro de la propiedad.

    5.- El sistema envía la notificación "orden registro de la propiedad" que envía el registro de la propiedad a la entidad de crédito para la confirmación de la inscripción en la BD.

Escenario Alternativo:

- 3-4.-a) La variante del acuerdo es directo:

  1.- El sistema recibe un mensaje del usuario con el EGRID o número del inmueble y comprueba que el inmueble no existe en la base de datos y envía un mensaje con la confirmación.

  Se repite el paso 1 hasta que no haya más inmuebles.

- 4.-a) El inmueble ya existe.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 6.-a) El inmueble no está en el mismo distrito del registro de la propiedad.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 7.-a) La variante declaración de acuerdo del propietario es establecida directamente.

  1.- El sistema recibe mediante un mensaje la declaración de acuerdo del propietario. Por medio del formulario "Declaración transformación de una cédula hipotecaria de registro" y registra la fecha de la firma en la BD.

  2.- El sistema añade el número de transacción enviándolo a través de un mensaje al formulario envío de notificación que completará la entidad de crédito.

- 11.-a) La hipoteca ya existe,

  1.- La oficina del registro de la propiedad enviará a través del sistema mediante una notificación del rechazo con todos los documentos de partida a la entidad de crédito.

  2.- Finaliza la ejecución del programa.

- 12.-a) Los datos a registrar no son correctos y la transacción ya existe,

  1.- La oficina del registro de la propiedad enviará a través del sistema mediante una notificación de rechazo con todos los documentos de partida a la entidad de crédito.

  2.- Finaliza la ejecución del programa.

- *.-a) El usuario puede cancelar el proceso en cualquier momento.

Caso de uso: Cambio de acreedor

También, conocido como "Recompra de créditos entre entidades de crédito".

Identificación de los escenarios:

- **Escenario Principal**: Cambio Acreedor OK(Variante a: Registrar cédula hipotecaria)
- **Escenario Alternativo 1 éxito**: Cambio Acreedor OK (Variante c: La cédula hipotecaria de registro ya está a disposición).
- **Escenario Alternativo 2 éxito**: La entidad de crédito vendedora (antigua) no está de acuerdo con el contenido de la promesa de pago.
- **Escenario de error 1**: El inmueble ya existe.
- **Escenario de error 2**: La entidad de crédito vendedora (antigua) no está registrado en la BD.
- **Escenario de error 3**: La entidad de crédito vendedora (antigua) no es la misma que la registrada en la BD.
- **Escenario de error 4**: Las entidades de crédito no pertenecen al mismo registro de la propiedad.
- **Escenario de error 5**: Uno o más plazos se pagan en más 18 meses.
- **Escenario de error 6**: El inmueble no está en el mismo distrito del registro de la propiedad.
- **Escenario de error 7**: La hipoteca ya existe.
- **Escenario de error 8**: Los datos a registrar no son correctos y la transacción ya existe.
- **Escenario de error 9**: La hipoteca no existe.
- **Escenario de excepción**: Cancelar transacción.

Actores: Entidad de crédito nueva (Actor principal), entidad de crédito antigua (Actor secundario) y oficina del registro de la propiedad (Actor secundario).

Descripción: Se realiza la transmisión de los derechos del acreedor sobre una cédula hipotecaria de registro, es efectuada por el antiguo acreedor por medio de un requerimiento de cesión ante la oficina del catastro afectado y la de papel es efectuada por el nuevo acreedor. La amortización de una hipoteca es en principio una cuestión de derecho privado que se realiza entre el tomador de crédito/prestatario (deudor hipotecario), la entidad de crédito vendedora (antigua) y la entidad de crédito compradora (nueva).

Precondiciones:

1. Deben ser utilizados cada vez que una entidad de crédito le rescata un crédito hipotecario a otra entidad de crédito.
2. La oficina del registro de la propiedad afectado está conectado al sistema.

3. La entidad de crédito compradora (nueva) y vendedora (antigua) son participantes del sistema, siendo bancos, los seguros y las cajas de pensiones.

4. El proceso es lanzado en principio por la entidad de crédito compradora (nueva).

5. Una transacción puede concernir varias cédulas hipotecarias y una cédula hipotecaria varios edificios, para que todas las cédulas hipotecarias y todos los edificios conciernan el mismo distrito del registro de la propiedad, es decir, la misma oficina del registro de la propiedad.

Postcondiciones: El sistema registra que una entidad de crédito a adquirido una cédula hipotecaria de otra entidad de crédito y puede procederse al cobro de las cuotas.

Escenario Principal:

1. El usuario de la entidad de crédito compradora (nueva) inicia el proceso e introduce la variante inicial.

2. El sistema recibe mediante un mensaje la variante introducida por el usuario.

3. El usuario introduce la variante del acuerdo.

4. El sistema recibe mediante un mensaje la variante del acuerdo introducida por el usuario.

5. El sistema recibe mediante un mensaje la declaración de acuerdo del propietario por medio del formulario “Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro” completado por el usuario.

6. El sistema recibe un mensaje del usuario con el EGRID o número del inmueble y comprueba que el inmueble no existe en la base de datos y envía un mensaje con la confirmación.

   Se repite el paso 6 hasta que no haya más inmuebles.

7. El usuario de la entidad de crédito añade los datos que faltan de la orden enviada por el sistema.

8. El sistema recibe mediante un mensaje la orden que le envía el usuario.

9. El sistema comprueba que las entidades de crédito vendedora y compradora pertenecen a un mismo registro de la propiedad.

10. El sistema recibe mediante un mensaje una promesa de pago condicional e irrevocable de la entidad de crédito compradora (nueva) a favor de la entidad de crédito vendedor (antigua) y está de acuerdo.

11. La entidad de crédito vendedora (antigua) confirma electrónicamente la promesa de pago mediante un mensaje que el sistema recibe.

12. El sistema envía la confirmación realizada por la entidad de crédito vendedora (antigua) a la entidad de crédito compradora (nueva) mediante un mensaje.

13. El sistema comprueba que uno o más plazos del crédito que se vuelve a comprar se deben pagar como máximo en 18 meses.

14. El sistema comprueba que el inmueble está en el mismo distrito que la oficina del registro de la propiedad.

    Se repite el paso 12 hasta que no haya más inmuebles.

15. La entidad de crédito vendedora (antigua) enviará en los plazos las garantías exigidas, la o las cédulas hipotecarias nominativas (papel) antes de ser endosadas y las pólizas de seguro notificadas.

16. La entidad de crédito vendedora confirma al sistema la recepción de las garantías para que el pago pueda ser posteriormente activado automáticamente.

17. La entidad de crédito compradora (nueva) confirma al sistema la recepción de las garantías necesarias.

18. El sistema añade el número de transacción enviándolo a través de un mensaje en el formulario (envío de notificación) a la entidad de crédito (nueva).

19. El usuario de la entidad de crédito (nueva) envía el formulario completado al sistema que lo recibe mediante un mensaje con los datos.

20. El sistema envía los documentos mediante un mensaje al registro de la propiedad (orden, declaración de acuerdo del propietario (con el nº de transacción) y envío de notificación (con el nº de transacción)).

21. El usuario de la oficina del registro de la propiedad recibe los documentos rellenados mediante un mensaje (cédula (orden), declaración (requisa) y envío de notificación) y añade los datos que faltan de la orden enviada por el sistema (EREID).

22. El sistema recibe mediante un mensaje la orden que le envía el usuario de la oficina del registro de la propiedad y aprueba que es correcta (Hipoteca no existe en la BD),

    1.- El sistema registra los datos electrónicos de la orden.

23. El sistema aprueba que los datos a registrar son correctos y la inscripción de la transacción en la BD (la transacción no existe en la BD):

    1.- El sistema registra la transacción en la base de datos.

    2.- La cédula hipotecaria en papel será anulada y destruida posteriormente.

    3.- La cédula hipotecaria en papel salió del control de títulos.

    4.- Se registró la cédula hipotecaria en el registro de la propiedad.

    5.- El sistema envía la notificación “orden registro de la propiedad” que envía el registro de la propiedad a las entidades de crédito implicadas para la confirmación de la inscripción en la BD.

Escenario Alternativo:

- 5-6.-a) La variante del acuerdo es directo:

  1.- El sistema recibe un mensaje del usuario con el EGRID o número del inmueble y comprueba que el inmueble no existe en la base de datos y envía un mensaje con la confirmación.

  Se repite el paso 1 hasta que no haya más inmuebles.

- 6.-a) El inmueble ya existe.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 9.-a) Las entidades de crédito no pertenecen al mismo registro de la propiedad.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 11.-a) La entidad de crédito vendedora (antigua) no está de acuerdo con el contenido de la promesa de pago.

  1.- No se llega a un acuerdo. La promesa de pago establecida es anulada.

  2.- La entidad de crédito vendedora (antigua) hace una contra propuesta.

3.- El sistema recibe mediante un mensaje una nueva promesa de pago condicional e irrevocable de la entidad de crédito compradora (nueva) a favor de la entidad de crédito vendedora (antigua) sobre la base de la contra propuesta.

11.-a) Repetir hasta que se llegue a un acuerdo.

- 13.-a) Uno o más plazos se pagan en más 18 meses.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 14.-a) El inmueble no está en el mismo distrito del registro de la propiedad.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 18.-a) La variante declaración de acuerdo del propietario es establecida directamente.

  1.- El sistema recibe mediante un mensaje la declaración de acuerdo del propietario. Por medio del formulario “Declaración transformación de una cédula hipotecaria de registro”.

  2.- El sistema añade el número de transacción enviándolo a través de un mensaje al formulario envío de notificación que completará la entidad de crédito.

- 22.-a) La hipoteca ya existe.

  1.- La oficina del registro de la propiedad enviará a través del sistema mediante una notificación del rechazo con todos los documentos de partida a la entidad de crédito (nueva).

  2.- Finaliza la ejecución del programa.

- 23.-a) Los datos a registrar no son correctos y la transacción ya existe.

  1.- La oficina del registro de la propiedad enviará a través del sistema mediante una notificación de rechazo con todos los documentos de partida a la entidad de crédito (nueva).

  2.- Finaliza la ejecución del programa.

- 3-23.-a) Variante cédula hipotecaria de registro ya a disposición.

  1.- El sistema comprueba el IDE de la entidad de crédito que financiaba hasta ahora es el de la entidad de crédito vendedora (Antigua) y que está registrada en la BD.

  2.- El sistema comprueba que las entidades de crédito vendedora y compradora pertenecen a un mismo registro de la propiedad.

  3.- El sistema recibe mediante un mensaje una promesa de pago condicional e irrevocable de la entidad de crédito compradora (nueva) a favor de la entidad de crédito vendedor (antigua).

  4.- La entidad de crédito vendedora (antigua) confirma electrónicamente la promesa de pago mediante un mensaje que el sistema recibe.

  5.- El sistema envía la confirmación realizada por la entidad de crédito vendedora (antigua) a la entidad de crédito compradora (nueva) mediante un mensaje.

  6.- El sistema comprueba que uno o más plazos del crédito que se vuelve a comprar se deben pagar como máximo en 18 meses.

  7.- La entidad de crédito vendedora (antigua) enviará en los plazos las garantías exigidas, la o las cédulas hipotecarias nominativas (papel) antes de ser endosadas y las pólizas de seguro notificadas.

  8.- La entidad de crédito vendedor confirma al sistema la recepción de las garantías para que el pago pueda ser posteriormente activado automáticamente.

9.- La entidad de crédito compradora (nueva) confirma al sistema la recepción de las garantías necesarias.

10.- El sistema recibe mediante un mensaje el formulario de cambio de acreedor por parte de la entidad de crédito (antigua).

11.- El sistema envía el formulario de cambio de acreedor al registro de la propiedad y la promesa de pago.

12.- El usuario del registro de la propiedad recibe los documentos mediante un mensaje.

13.- El usuario de la oficina del registro de la propiedad añade los datos que faltan del formulario de cambio de acreedor enviada por el sistema (EREID).

14.-El sistema recibe mediante un mensaje el formulario de cambio de acreedor que le envía el usuario de la oficina del registro de la propiedad y aprueba que es correcta (Hipoteca existe en la BD),

1. El sistema registra los datos electrónicos del formulario.

15.- El sistema aprueba que los datos a registrar son correctos y la inscripción de la transacción en la BD (la transacción no existe en la BD):

1. El sistema registra la transacción en la base de datos.
2. La oficina del registro de la propiedad confirma la inscripción en la BD a través del sistema mediante un mensaje de notificación a las entidades de crédito implicadas.

- 3-23.-a) 1.- a) La entidad de crédito vendedora (antigua) no está registrado en la BD.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 3-23.-a) 1.- b) La entidad de crédito que garantiza la financiación hasta ahora que no corresponde con la entidad de crédito vendedora (antigua).

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 3-23.-a) 2.- a) Las entidades de crédito no pertenecen al mismo registro de la propiedad.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 3-23.-a) 4.- a) La entidad de crédito vendedora (antigua) no está de acuerdo con el contenido de la promesa de pago.

  1.- No se llega a un acuerdo. La promesa de pago establecida es anulada.

  2.- La entidad de crédito vendedora (antigua) hace una contra propuesta.

  3.- El sistema recibe mediante un mensaje una nueva promesa de pago condicional e irrevocable de la entidad de crédito compradora (nueva) a favor de la entidad de crédito vendedora (antigua) sobre la base de la contra propuesta.

  Repetir paso 4 hasta que se llegue a un acuerdo.

- 3-23.-a) 6.- a) Uno o más plazos se pagan en más 18 meses.

  1.- El sistema muestra un mensaje indicando el error producido y finaliza la ejecución del programa.

- 3-23.-a) 14.- a) La hipoteca no existe.

  1.- La oficina del registro de la propiedad enviará a través del sistema mediante una notificación del rechazo con todos los documentos de partida a la entidad de crédito.

  2.- Finaliza la ejecución del programa.

- 3-23.-a) 15.- a) Los datos a registrar no son correctos y la transacción ya existe.

  1.- La oficina del registro de la propiedad enviará a través del sistema mediante una notificación de rechazo con todos los documentos de partida a la entidad de crédito.

  2.- Finaliza la ejecución del programa.

- *.-a) El usuario puede cancelar el proceso en cualquier momento.

Caso de uso: Constitución de cédulas hipotecarias de registro

Identificación de los escenarios:

- **Escenario Principal**: Constitución OK.
- **Escenario de error 1**: La hipoteca no existe.
- **Escenario de error 2**: Los datos a registrar no son correctos y la transacción ya existe.
- **Escenario de excepción**: Cancelar transacción.

Actores: Entidad de crédito (Actor principal), notario (Actor secundario) y oficina del registro de la propiedad (Actor secundario).

Descripción: La constitución de la cédula hipotecaria de registro se hará bien mediante el registro de un contrato de garantía de dos páginas o de una declaración de una página. Es posible constituir una cédula hipotecaria de registro por contrato de garantía.

Precondiciones: La cédula hipotecaria tiene que ser de registro para este proceso.

Postcondiciones: El sistema ha constituido las cédulas hipotecarias de registro almacenando el contrato de garantía.

Escenario Principal:

1. El usuario de la entidad de crédito inicia el proceso.
2. El sistema recibe mediante un mensaje el formulario con los datos para la constitución de la cédula hipotecaria de registro por parte de la entidad de crédito.
3. El sistema envía la orden al notario mediante un mensaje y este lo recibe junto con los datos.
4. El propietario contacta con el notario, recibe por su parte un código de activación y fijará una cita.
5. El notario descarga la orden de la entidad de crédito directamente en un modelo de contrato de garantía e imprime este último.
6. Una vez los trámites terminados, el propietario firmará el contrato de garantía con la firma del notario y el contrato es escaneado en Portable Document Format (PDF) por el notario.
7. El notario confirma poniendo una firma digital notarial sobre el documento PDF (contrato de garantía escaneado).

8. El notario rellena lo que falta en el formulario "Requisa electrónica creación de una cédula hipotecaria de registro" (Introduce el código de activación y adjunta PDF que es el contrato de garantía escaneado) y se lo vuelve a enviar al sistema.

9. El sistema le envía la requisa a la oficina del registro de la propiedad.

10. El usuario de la oficina del registro de la propiedad añade los datos que faltan del formulario enviada por el sistema (EREID).

11. El sistema recibe mediante un mensaje el formulario que le envía el usuario de la oficina del registro de la propiedad y aprueba que es correcto (La hipoteca está registrada en la BD),

    1.- La cédula hipotecaria de registro se incluye en la BD.

12. El sistema aprueba que los datos a registrar son correctos y la inscripción de la transacción en la BD (la transacción no existe en la BD):

    1.- El sistema registra la transacción en la BD.

    2.- La oficina del registro de la propiedad confirma la inscripción en la BD a través del sistema mediante un mensaje de notificación con la confirmación electrónica de la inscripción de la cédula hipotecaria de registro a la entidad de crédito implicada.

Escenario Alternativo:

- 11.-a) La hipoteca no existe.

  1.- La oficina del registro de la propiedad enviará a través del sistema mediante una notificación de rechazo a la entidad de crédito.

  2.- Finaliza la ejecución del programa.

- 12.-a) Los datos a registrar no son correctos y la transacción ya existe.

  1.- La oficina del registro de la propiedad enviará a través del sistema mediante una notificación de rechazo a la entidad de crédito.

  2.- Finaliza la ejecución del programa.

- *.-a) El usuario puede cancelar el proceso en cualquier momento.

3.5. Modelo conceptual de datos

El modelo conceptual de datos [35] es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Utilizando la notación UML, un modelo conceptual de datos se representa con un conjunto de diagrama de clases en los que no se define ninguna operación. Puede mostrar:

- Objetos del dominio o clases conceptuales. Describe un conjunto de objetos con las mismas propiedades, comportamiento común, idéntica relación con otros objetos y semántica común.
- Asociaciones entre las clases conceptuales. Es la representación de relaciones entre dos o más objetos.
- Atributos de las clases conceptuales. Es una propiedad compartida por los objetos de una clase.
- Restricciones de integridad. Limitan los valores que pueden tomar las clases, los atributos o las asociaciones.

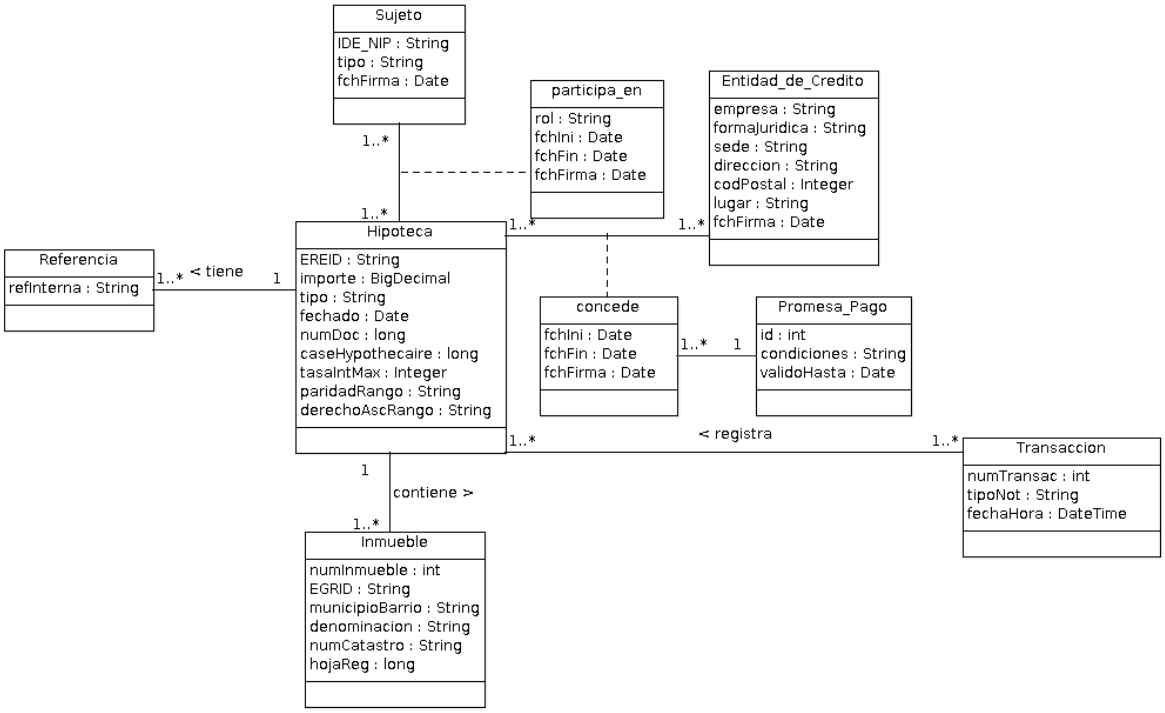


Figura 3.2.: Modelo conceptual de datos (Hipoteca)

El modelo conceptual de datos de este proyecto se ha dividido en diferentes figuras porque es demasiado extenso y se va a mostrar mejor de forma detallada. Primero, se puede observar en la figura 3.2 el primer diagrama que contiene la parte genérica de este modelo porque Hipoteca es la que se relaciona con un mayor número de clases. En las hipotecas participan un conjunto de sujetos y las entidades de crédito conceden un conjunto de hipotecas mediante una o ninguna promesa de pago. La hipoteca tiene una referencia y contiene un conjunto de inmuebles. Finalmente, las transacciones registran un conjunto de hipotecas.

En la figura 3.3 se puede observar un diagrama más específico, dedicado exclusivamente a los sujetos, estos pudiendo ser, comunidades, personas físicas y jurídicas. Una comunidad está formada por un conjunto de personas físicas. En la figura 3.4 se puede observar otro diagrama específico que indica que cada transacción guarda un conjunto de documentos o puede dar el caso que no se guarde ninguno. La figura 3.5 muestra un nuevo diagrama dedicado a las clases que se relacionan con la referencia. Es seguida por un usuario y firmada por 0, 1 o 2 usuarios que realizan una determinada función. Los usuarios se registran con un rol en la base de datos.

En la figura 3.6 se puede observar otro diagrama específico que indica que las entidades realizan un conjunto de transacciones cuyos roles son expedidor y destinatario. Los tres tipos de entidades son las entidades de crédito, los registros de la propiedad y los notarios. Finalmente, el registro de la propiedad se relaciona con un conjunto de entidades de crédito y realiza la transacción. Para finalizar, se completa el modelo conceptual de datos con el último diagrama sobre las promesas de pago como se observa en la figura 3.7. Una promesa de pago está compuesta por un conjunto de plazos.

3.6. Modelo de comportamiento del sistema

La especificación del comportamiento se hace con el modelo de comportamiento del sistema. El comportamiento del sistema [35] es una descripción de qué hace el sistema, sin explicar cómo lo hace. Se han de

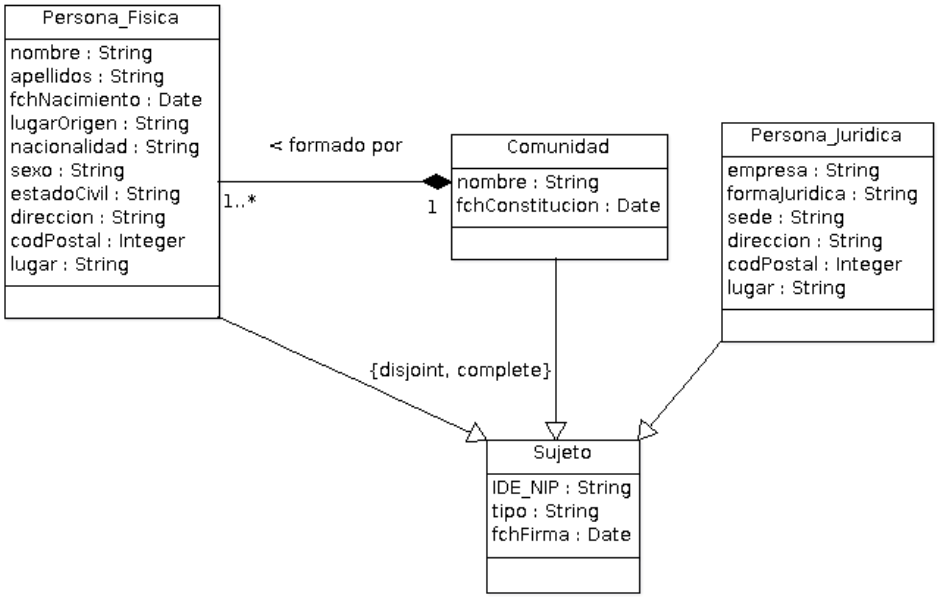


Figura 3.3.: Modelo conceptual de datos (Sujeto)

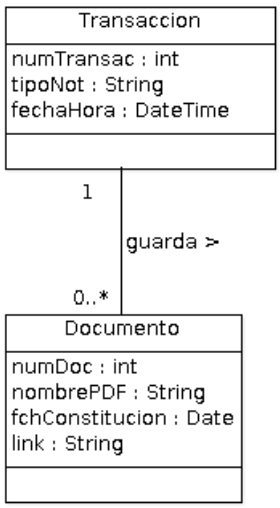


Figura 3.4.: Modelo conceptual de datos (Transaccion)

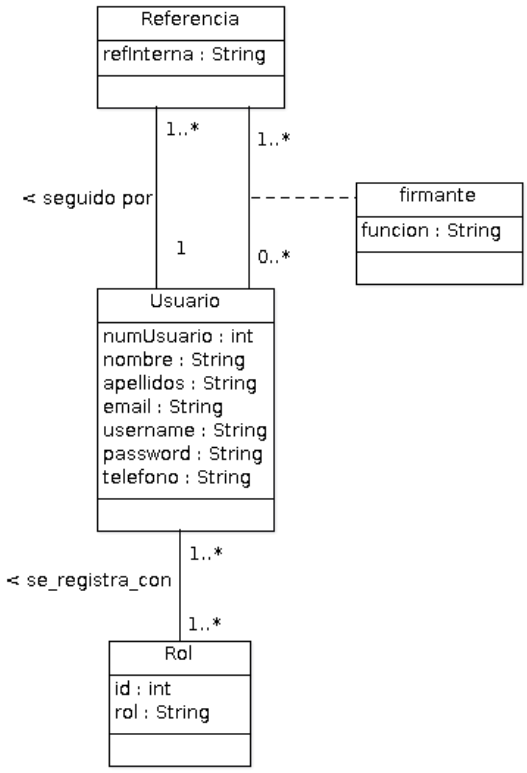


Figura 3.5.: Modelo conceptual de datos (Referencia)

realizar para llevar a cabo dicho modelo: los diagramas de secuencia y los contratos de las operaciones. Se aplican a los casos de usos descritos en el apartado "Especificación de casos de uso".

Un Diagrama de Secuencia del Sistema (DSS) es un dibujo que muestra, para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y los eventos entre los sistemas. Los contratos de las operaciones describen el efecto de las operaciones del sistema.

3.6.1. Caso de uso: «Registrar cédula hipotecaria»

Diagrama de comportamiento

Se puede observar en la figura 3.8 el diagrama de comportamiento del caso de uso: Registrar cédula hipotecaria.

Contrato de las operaciones

Operación: InsertarDatosFormularios(w_datos).

Responsabilidades: Se encarga de introducir los datos de los siguientes formularios: "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" en la que se autoriza que el registro de la propiedad destruya la cédula en papel, "Orden Transformación Cédula hipotecaria" en la que se rellenan los datos a registrar y "Envío de notificación sobre la transformación de una hipoteca" en la que se indica si se realiza de forma electrónica o por papel entre otros datos. Además, se validan los inmuebles introducidos y se comprueban que estén en el mismo distrito que el registro de la propiedad.

Referencias cruzadas: Caso de uso: Registrar cédula hipotecaria. Véase 3.4.2.

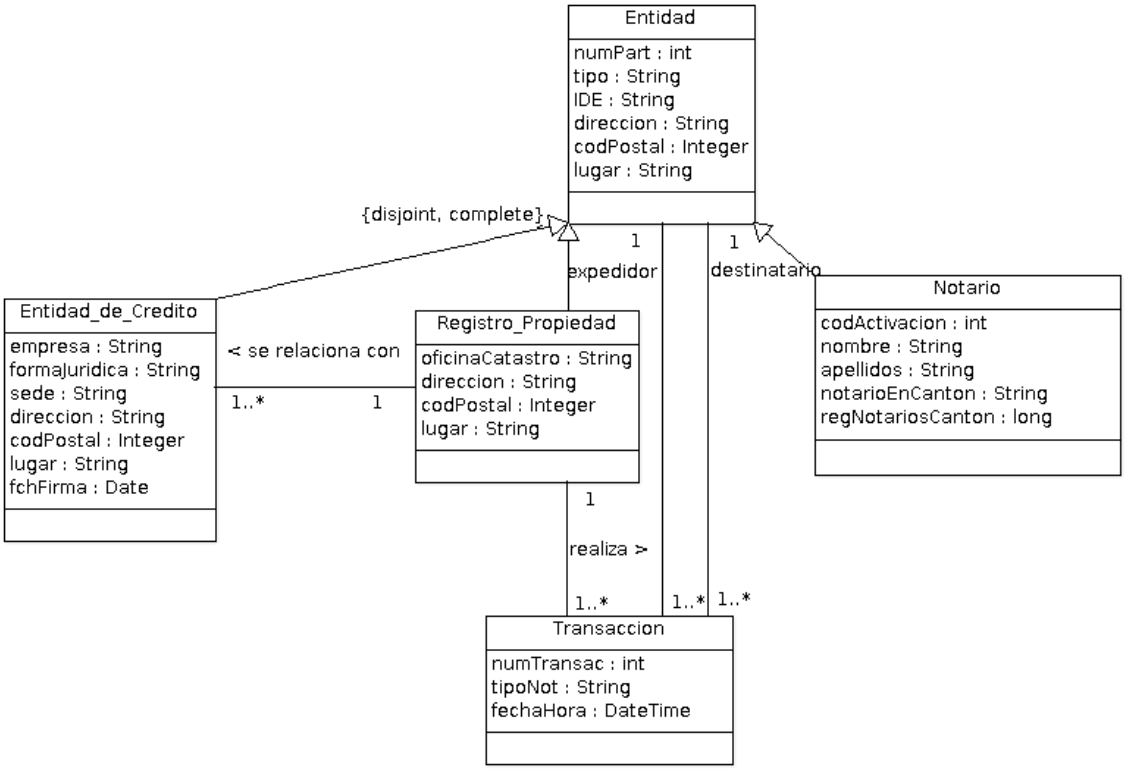


Figura 3.6.: Modelo conceptual de datos (Entidad)

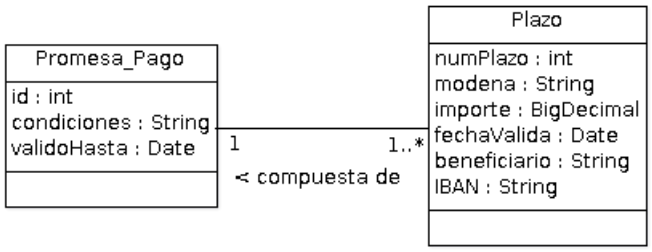


Figura 3.7.: Modelo conceptual de datos (Promesa de pago)

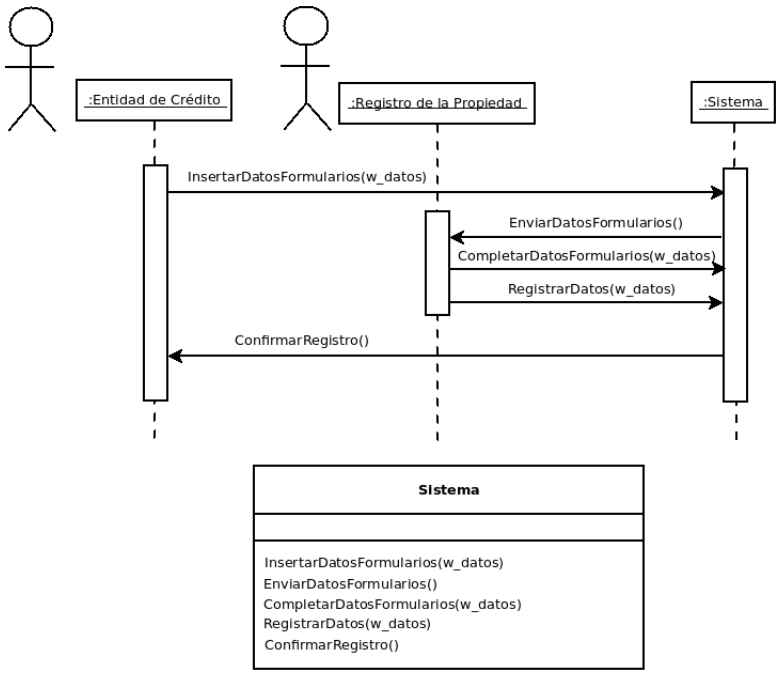


Figura 3.8.: Modelo de Comportamiento del caso de uso: Registrar cédula hipotecaria.

Precondiciones:

- No existe un Inmueble cuyo EGRID o cuyo numInmueble sea igual al introducido.
- Existe un Registro_Propiedad cuyo numPart es igual al introducido.
- Existe un lugar cuyo codPostal es igual al introducido.

Postcondiciones: El sistema obtiene los datos introducidos por el usuario de la entidad de crédito.

Operación: EnviarDatosFormularios().

Responsabilidades: Se encarga de enviar los datos introducidos a la oficina del registro de la propiedad.

Referencias cruzadas: Caso de uso: Registrar cédula hipotecaria. Véase 3.4.2.

Precondiciones: Que se hayan introducido los datos previamente.

Postcondiciones: La oficina del registro de la propiedad recibe los datos introducidos por el usuario de la entidad de crédito.

Operación: CompletarDatosFormularios(w_datos).

Responsabilidades: Se encarga de introducir los datos que faltan en el formulario "Orden Transformación Cédula hipotecaria" porque deben ser introducidos por el usuario de la oficina del registro de la propiedad y este formulario pasa a llamarse "Confirmación registrado en el registro de la propiedad".

Referencias cruzadas: Caso de uso: Registrar cédula hipotecaria. Véase 3.4.2.

Precondiciones: Que se hayan recibido los formularios.

Postcondiciones: El sistema obtiene los datos que deben ser introducidos por el usuario de la oficina del registro de la propiedad.

Operación: RegistrarDatos(w_datos).

Responsabilidades: Se comprueba si es aprobado o no el registro de los datos de la hipoteca y si es aprobado, se crea todas las instancias necesarias para el registro de los datos del formulario "Orden Transformación Cédula hipotecaria" sobre la hipoteca. Además, se comprueba si es el contenido de los datos del formulario "Orden Transformación Cédula hipotecaria" y si se puede registrar los datos de la transacción, si es así, se crean todas las instancias necesarias para el registro de la transacción.

Referencias cruzadas: Caso de uso: Registrar cédula hipotecaria. Véase 3.4.2.

Precondiciones:

- No existe una Hipoteca cuyo EREID sea igual al introducido.
- No existe una Transaccion cuyo numTransac sea igual al introducido.

Postcondiciones: Se crean cada una de las instancias necesarias para la creación de los objetos (Hipoteca, AsignarAcreedor, SujetoHipoteca, Inmueble, Referencia, Firmante, Transaccion, HipotecaTransaccion). Se le asignan los datos a los atributos de dichos objetos (modificación de atributos). Finalmente, se crean los enlaces necesarios.

Operación: ConfirmarRegistro().

Responsabilidades: Se envía la notificación (formulario) para confirmar a la entidad de crédito que se ha realizado correctamente el registro.

Referencias cruzadas: Caso de uso: Registrar cédula hipotecaria. Véase 3.4.2.

Precondiciones: Que se haya realizado el registro.

Postcondiciones: La entidad de crédito recibe la notificación confirmando el registro.

3.6.2. Caso de uso: «Cambio de acreedor»

También, conocido como "Recompra de créditos entre entidades de crédito".

Diagrama de comportamiento

Se puede observar en la figura 3.9 el diagrama de comportamiento del caso de uso: Cambio de acreedor.

Contrato de las operaciones

Operación: InsertarDatosFormularios(w_datos).

Responsabilidades: Se encarga de introducir los datos de los siguientes formularios: "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" en la que se autoriza que el registro de la propiedad destruya la cédula en papel, "Orden Transformación Cédula hipotecaria" en la que se rellenan los datos a registrar y "Envío de notificación sobre la transformación de una hipoteca" en la que se indica si se realiza de forma electrónica o por papel entre otros datos. Además, se validan los inmuebles introducidos y se comprueban que estén en el mismo distrito que el registro de la propiedad.

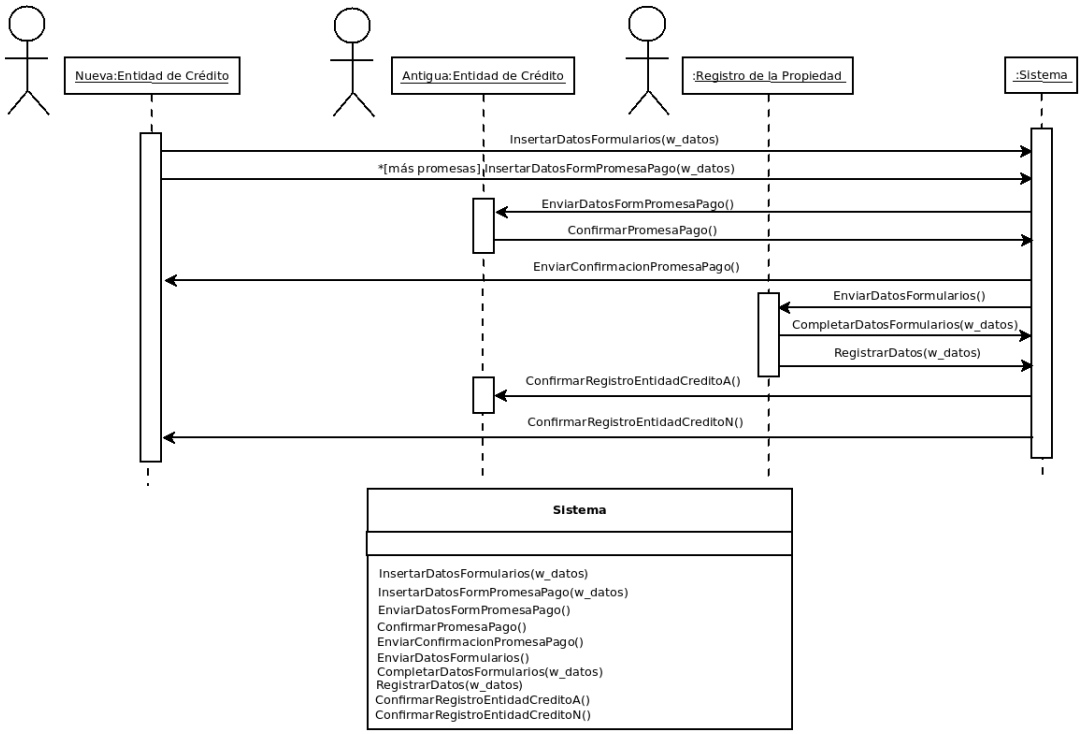


Figura 3.9.: Modelo de Comportamiento del caso de uso: Cambio de acreedor.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones:

- No existe un Inmueble cuyo EGRID o cuyo numInmueble sea igual al introducido.
- Existe un Registro_Propiedad cuyo numPart es igual al introducido.
- Existe un lugar cuyo codPostal es igual al introducido.

Postcondiciones: El sistema obtiene los datos introducidos por el usuario de la entidad de crédito nueva.

Operación: InsertarDatosFormPromesaPago(w_datos).

Responsabilidades: Se encarga de introducir los datos del formulario “Promesa de pago condicional e irrevocable”.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones: Ninguna.

Postcondiciones: El sistema obtiene los datos introducidos por el usuario de la entidad de crédito sobre la promesa de pago.

Operación: EnviarDatosFormPromesaPago().

Responsabilidades: Se encarga de enviar los datos introducidos a la entidad de crédito antigua.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones: Que se hayan introducido los datos previamente.

Postcondiciones: La entidad de crédito antigua recibe los datos introducidos por el usuario de la entidad de crédito nueva sobre la promesa de pago.

Operación: ConfirmarPromesaPago().

Responsabilidades: Se confirma que se aceptan las condiciones de la promesa de pago.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones: Que se hayan introducido los datos previamente.

Postcondiciones: El sistema recibe la confirmación de la promesa de pago por el usuario de la entidad de crédito antigua.

Operación: EnviarConfimacionPromesaPago().

Responsabilidades: Se encarga de enviar la confirmación a la entidad de crédito nueva.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones: Que se haya confirmado la promesa de pago.

Postcondiciones: La entidad de crédito nueva recibe la confirmación de la promesa de pago del usuario de la entidad de crédito antigua.

Operación: EnviarDatosFormularios().

Responsabilidades: Se encarga de enviar los datos introducidos a la oficina del registro de la propiedad.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones: Que se hayan introducido los datos previamente.

Postcondiciones: La oficina del registro de la propiedad recibe los datos introducidos por el usuario de la entidad de crédito nueva.

Operación: CompletarDatosFormularios(w_datos).

Responsabilidades: Se encarga de introducir los datos que faltan en el formulario "Orden Transformación Cédula hipotecaria" porque deben ser introducidos por el usuario de la oficina del registro de la propiedad y este formulario pasa a llamarse "Confirmación registrado en el registro de la propiedad".

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones: Que se hayan recibido los formularios.

Postcondiciones: El sistema obtiene los datos que deben ser introducidos por el usuario de la oficina del registro de la propiedad.

Operación: RegistrarDatos(w_datos).

Responsabilidades: Se comprueba si es aprobado o no el registro de los datos de la hipoteca y si es aprobado, se crea todas las instancias necesarias para el registro de los datos del formulario "Orden Transformación Cédula hipotecaria" sobre la hipoteca. Además, se comprueba si es el contenido de los datos del formulario "Orden Transformación Cédula hipotecaria" y si se puede registrar los datos de la transacción, si es así, se crean todas las instancias necesarias para el registro de la transacción. También, se crean las instancias necesarias para el registro de la promesa de pago con sus plazos.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones:

- No existe una Hipoteca cuyo EREID sea igual al introducido.
- No existe una Transaccion cuyo numTransac sea igual al introducido.

Postcondiciones: Se crean cada una de las instancias necesarias para la creación de los objetos (Hipoteca, AsignarAcreedor, SujetoHipoteca, Inmueble, Referencia, Firmante, Transaccion, HipotecaTransaccion, PromesaPago, Plazo). Se le asignan los datos a los atributos de dichos objetos (modificación de atributos). Finalmente, se crean los enlaces necesarios.

Operación: ConfirmarRegistroEntidadCreditoA().

Responsabilidades: Se envía la notificación (formulario) para confirmar a la entidad de crédito antigua que se ha realizado correctamente el registro.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones: Que se haya realizado el registro.

Postcondiciones: La entidad de crédito antigua recibe la notificación confirmando el registro.

Operación: ConfirmarRegistroEntidadCreditoN().

Responsabilidades: Se envía la notificación (formulario) para confirmar a la entidad de crédito nueva que se ha realizado correctamente el registro.

Referencias cruzadas: Caso de uso: Cambio de acreedor. Véase 3.4.2.

Precondiciones: Que se haya realizado el registro.

Postcondiciones: La entidad de crédito nueva recibe la notificación confirmando el registro.

3.6.3. Caso de uso: «Constitución de cédulas hipotecarias de registro»

Diagrama de comportamiento

Se puede observar en la figura 3.10 el diagrama de comportamiento del caso de uso: Constitución de cédulas hipotecarias de registro.

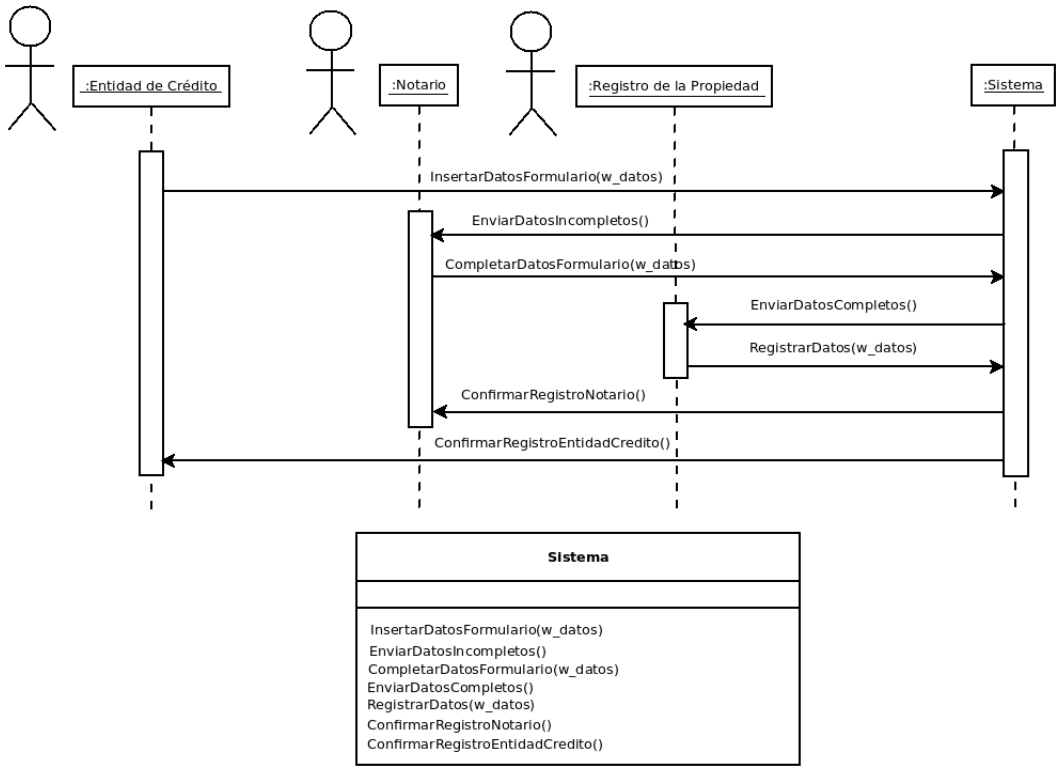


Figura 3.10.: Modelo de Comportamiento del caso de uso: Constitución de cédulas hipotecarias de registro.

Contrato de las operaciones

Operación: InsertarDatosFormulario(w_datos).

Responsabilidades: Se encarga de introducir los datos del formulario "Orden creación de una cédula de registro".

Referencias cruzadas: Caso de uso: Constitución de cédulas hipotecarias de registro. Véase 3.4.2.

Precondiciones: Ninguna.

Postcondiciones: El sistema obtiene los datos introducidos por el usuario de la entidad de crédito.

Operación: EnviarDatosIncompletos().

Responsabilidades: Se encarga de enviar los datos introducidos al notario.

Referencias cruzadas: Caso de uso: Constitución de cédulas hipotecarias de registro. Véase 3.4.2.

Precondiciones: Que se hayan introducido los datos previamente.

Postcondiciones: El notario recibe los datos introducidos por el usuario de la entidad de crédito.

Operación: CompletarDatosFormulario(w_datos).

Responsabilidades: Se encarga de introducir los datos que faltan en el formulario "Orden creación de una cédula de registro" porque deben ser introducidos por el notario y este formulario pasa a llamarse "Requisa creación de una cédula de registro".

Referencias cruzadas: Caso de uso: Constitución de cédulas hipotecarias de registro. Véase 3.4.2.

Precondiciones: Que se hayan recibido el formulario.

Postcondiciones: El sistema obtiene los datos que deben ser introducidos por el usuario del notario.

Operación: EnviarDatosCompletos().

Responsabilidades: Se encarga de enviar los datos introducidos a la oficina del registro de la propiedad.

Referencias cruzadas: Caso de uso: Constitución de cédulas hipotecarias de registro. Véase 3.4.2.

Precondiciones: Que se hayan introducido los datos previamente.

Postcondiciones: La oficina del registro de la propiedad recibe los datos introducidos por el usuario de la entidad de crédito.

Operación: RegistrarDatos(w_datos).

Responsabilidades: Se comprueba si es aprobado o no el registro de los datos de la hipoteca y si es aprobado, se crea todas las instancias necesarias para el registro de los datos del formulario "Requisa creación de una cédula de registro" sobre la hipoteca. Además, se comprueba si es el contenido de los datos del formulario "Requisa creación de una cédula de registro" y si se puede registrar los datos de la transacción, si es así, se crean todas las instancias necesarias para el registro de la transacción.

Referencias cruzadas: Caso de uso: Constitución de cédulas hipotecarias de registro. Véase 3.4.2.

Precondiciones:

- Existe una Hipoteca cuyo EREID sea igual al introducido.
- No existe una Transaccion cuyo numTransac sea igual al introducido.

Postcondiciones: Se crean cada una de las instancias necesarias para la creación de los objetos (Hipoteca, Transaccion, HipotecaTransaccion, Documento). Se le asignan los datos a los atributos de dichos objetos (modificación de atributos). Finalmente, se crean los enlaces necesarios.

Operación: ConfirmarRegistroNotario().

Responsabilidades: Se envía la notificación (formulario) para confirmar al notario que se ha realizado correctamente el registro.

Referencias cruzadas: Caso de uso: Constitución de cédulas hipotecarias de registro. Véase 3.4.2.

Precondiciones: Que se haya realizado el registro.

Postcondiciones: El notario recibe la notificación confirmando el registro.

Operación: ConfirmarEntidadCredito().

Responsabilidades: Se envía la notificación (formulario) para confirmar a la entidad de crédito que se ha realizado correctamente el registro.

Referencias cruzadas: Caso de uso: Constitución de cédulas hipotecarias de registro. Véase 3.4.2.

Precondiciones: Que se haya realizado el registro.

Postcondiciones: La entidad de crédito recibe la notificación confirmando el registro.

4. Diseño del proyecto

La etapa de diseño es la etapa central en relación con la arquitectura y probablemente la más compleja. Durante esta etapa se definen las estructuras que componen la arquitectura. El diseño que se realiza debe buscar ante todo satisfacer los requerimientos que influencian a la arquitectura. A continuación, se definen los pasos a seguir para el desarrollo de la base de datos, pero todo sin ver detalles de la implementación.

4.1. Arquitectura del sistema

La arquitectura de software de un sistema [37] es el conjunto de estructuras necesarias del sistema, que incluyen los componentes del software, las relaciones entre ellos, y propiedades de ambos. Tener una arquitectura del software es importante para el éxito del desarrollo de un sistema de software.

Uno de los motivos para determinar la arquitectura del sistema ha sido la búsqueda de la calidad del software: el objetivo es crear un software flexible, mantenible, reutilizable y que sea seguro. Este proyecto se ha diseñado como una combinación de tipos de arquitecturas que conforman el sistema. Se tiene un diseño SOA utilizando un enfoque de arquitectura en capas y un estilo de arquitectura orientada a objetos. La arquitectura orientada a servicios, describe una colección de componentes distribuidos que proporcionan y / o consumen servicios. A continuación, se detallan las distintas arquitecturas.

4.1.1. Arquitectura en capas

Los módulos de esta estructura se denominan capas. Cada capa es una agrupación de módulos que ofrece un conjunto de servicios (cohesivos entre sí). La idea de hacer una separación en capas es que cada una de las mismas cumpla con un rol y tenga responsabilidades bien definidas. El principal objetivo es la separación de la lógica de negocios de la capa lógica de diseño. La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada. La arquitectura en capas se centra en la agrupación de funcionalidad relacionada dentro de una aplicación en distintas capas que se apilan verticalmente en la parte superior de la otra. Esto se hace con la intención de elevar el nivel de abstracción.

Los componentes de una capa pueden interactuar sólo con componentes de la misma capa o con componentes de la capa inmediatamente inferior. Las capas se comunican a través de interfaces y las implementaciones están ocultas al exterior. Las capas de una aplicación pueden residir en el mismo equipo físico o pueden estar distribuidos en equipos independientes, y los componentes de cada capa comunicarse con componentes de otras capas a través de interfaces bien definidas. Los principales beneficios del estilo de arquitectura basado en capas son:

- **Abstracción.** Las capas permiten cambios que se realicen en un nivel abstracto.
- **Aislamiento.** Permite aislar los cambios en tecnologías a ciertas capas para reducir el impacto en el sistema total.
- **Rendimiento.** Distribuir las capas entre múltiples sistemas (físicos) puede incrementar la escalabilidad, la tolerancia a fallos y el rendimiento.

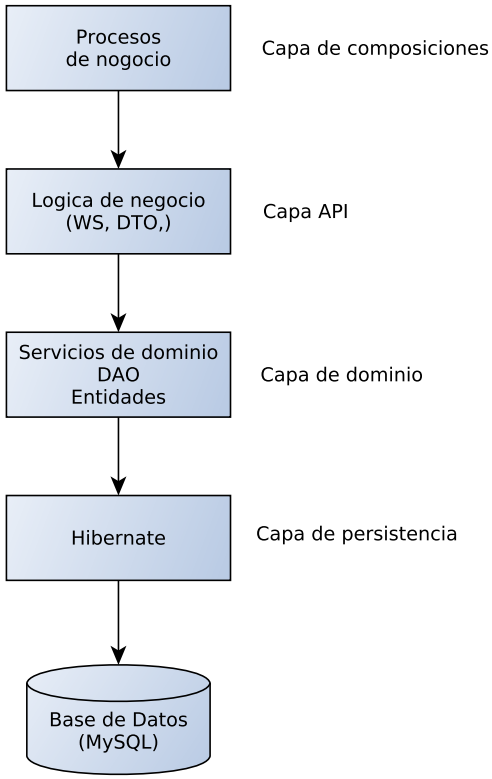


Figura 4.1.: Arquitectura en capas.

- **Mejoras en pruebas.** La capacidad de realizar pruebas se beneficia de tener unas interfaces bien definidas para cada capa así como de la habilidad para cambiar a diferentes implementaciones de las interfaces de cada capa.
- **Independencia.** El requerimiento de considerar el hardware y los problemas de instalación así como las dependencias de interfaces externas.
- **Reusabilidad.** Las capas inferiores no tienen dependencias de las superiores por lo que es fácil reutilizarlas.

Las capas de esta arquitectura se muestran en la figura 4.1. A continuación, se describen dichas capas, iniciando desde lo más bajo hacia arriba.

Capa de persistencia

El nivel de aplicación que se ocupa de la persistencia a menudo se llama capa de persistencia [38], responsabilidad de persistir los datos en la base de datos. La necesidad de vincular los datos guardados en una base de datos relacional, con los objetos de una aplicación orientada a objetos, determinó la aparición del concepto de persistencia de objetos. Los objetos persistentes se almacenan en la base de datos y persisten después de la terminación del programa.

Capa de dominio

La capa de dominio se divide en las siguientes capas [38], detalladas de abajo hacia arriba para comprender mejor esta estructura interna:

- **Capa de entidades.** Representa las entidades dentro de una aplicación, definen la forma en que se relacionan entre sí. Cada entidad define una serie de propiedades, que designa a sus características, así como sus relaciones con otras entidades. Cada clase en el modelo de dominio contiene las diversas propiedades y asociaciones que se correlacionan con las columnas y las relaciones dentro de la base de datos. Cada una de estas clases son objetos de negocio o Business Object (BO).
- **Capa Data Access Object (DAO).** La capa DAO define los medios para guardar y consultar los datos del modelo de dominio. DAO hace referencia a un patrón de diseño de la capa de persistencia que se encarga de cumplir con el principio Interface Segregation Principle (ISP) y separar las responsabilidades de negocio y persistencia. Con frecuencia un DAO para cada entidad del modelo y son obtenidos a través de una factoría. DAO proporciona una interfaz única de acceso a los datos, de forma independiente a dónde se hallen almacenados. Independiza la lógica de negocio del acceso a los datos. Ofrece operaciones CRUD para cada objeto persistente del dominio.
- **Capa de servicios de dominio.** Se debía simplificar el acceso a la capa de persistencia. La capa de servicios normalmente define Application Programming Interface (API) de cara al público de una aplicación, la combinación de una o más operaciones de DAO de menor nivel en una sola unidad transaccional cohesionada. La capa que se encarga de la lógica de negocio de la aplicación es la capa de servicios de dominio.

Las definiciones de algunos principios del diseño orientado a objeto cumplidos son [39]:

> "ISP: Una clase cliente A que tiene una dependencia con la clase B no debe verse forzada a depender de métodos de la clase B que no vaya a usar jamás."
>
> "Don't Repeat Yourself (DRY): Implica que, cualquier funcionalidad existente en un programa debe existir de forma única en él, o lo que es lo mismo, no debemos tener bloques de código repetidos."
>
> "Inversion of Control (IOC): Consiste en que el control de la construcción de los objetos no recae directamente en el desarrollador a través del uso del operador «new», sino que es otra clase o conjunto de clases las que se encargan de construir los objetos que necesitamos."
>
> "Dependency Injection Principle (DIP): Las dependencias que una clase tiene no deben ser asignadas por ella misma sino por un agente externo."
>
> "Convention Over Configuration (COC): Define que, antes de abordar el desarrollo, un programador puede declarar una serie de convenciones que le permiten asumir una configuración por defecto del sistema."

Capa API

Esta capa, representa la lógica de negocio del sistema, está formada por los servicios de negocio, los cuales, algunos son necesarios al ser llamados por las composiciones de la capa superior para completar su diseño. La capa API se divide a su vez en varias subcapas según el tipo de servicio. Véase 4.1.2.

Capa de composiciones

Es la capa más alta, representa los procesos de negocio del sistema, son composiciones de servicios. En esta capa se ubican los servicios compuestos, formados por la combinación de servicios más sencillos de la capa anterior. Se encuentran los procesos Business Process Management (BPM) que modelan casos de uso específicos y procesos de negocio de la organización, se usan coreografías u orquestación.

Un proceso de negocio es definido como [1]:

> "A business process is a set of coordinated activities that are performed either by humans or by tools with an objective to realize a certain business result."

Un proceso de negocio consiste en un conjunto de actividades coordinadas que logran un objetivo de negocio en particular. Es en gran medida en el interés de todas las empresas contar con procesos de negocio que son eficientes e incluyen sólo las actividades necesarias, porque esto les permitirá trabajar con mayor rapidez.

El diseño de los procesos se realiza con tecnología BPM. La gestión de procesos de Negocio o BPM [1] es un método para alinear una organización empresarial con las necesidades de sus clientes. BPM fomenta la eficacia empresarial y la eficiencia mientras se esfuerza para la innovación, la flexibilidad y la integración con la tecnología. El principal objetivo de BPM es la mejora continua de los procesos, tanto dentro de la empresa y con otras empresas.

Los procesos de negocio son dinámicos, por lo tanto, para expresar las diversas etapas, el ciclo de vida del proceso de negocio tiene que cubrir los siguientes cuatro fases:

- **Modelado de procesos.** Fase en la que los analistas del proceso, junto con los propietarios de los procesos, analizan el proceso de negocio y definen el modelo de proceso. Ellos definen el flujo de actividad, el flujo de información, funciones y documentos de la empresa. También definen las políticas comerciales y las restricciones, reglas de negocio, y las medidas de rendimiento.

- **Implementación del proceso.** Fase en la que los desarrolladores de Information Technology (IT) (desarrolladores SOA), junto con los analistas de procesos, implementan los procesos de negocio con el objetivo de dar soporte de extremo a extremo para el proceso de utilización de las IT (aplicaciones). La fase de ejecución de proceso utilizando el enfoque de SOA incluye la implementación de procesos con BPEL y descomposición de procesos para los servicios, la identificación del servicio, la aplicación o la reutilización de los servicios, y la integración.

- **Ejecución y control de procesos.** Fase de ejecución propiamente dicha, donde los participantes del proceso ejecutan las distintas actividades del proceso. Una parte importante de esta fase es el control de procesos, donde los supervisores o gestores del proceso controlan si se está ejecutando de manera óptima. Si se producen retrasos, surgen excepciones, los recursos no están disponibles, o ocurren algunas otras anomalías, los supervisores o gerentes pueden tomar acciones correctivas.

- **Monitorización y optimización de procesos.** Fase final y muy importante. En esta fase, los propietarios de los procesos monitorean los key performance indicator (KPI) del proceso. Analistas, propietarios, supervisores y los principales usuarios examinan el proceso y analizan las métricas de ejecución de los procesos. También, tienen que tener en cuenta las condiciones cambiantes del negocio. Ellos examinan temas de negocios e identifican formas de mejorar los procesos de negocio para eliminar estos problemas.

En la figura 4.2 se muestra cómo un proceso entra en este círculo y pasa por las distintas etapas. Una vez se han identificado y seleccionado las optimizaciones, el proceso vuelve a la fase de modelado. Entonces el proceso se reimplementa, y todo el ciclo de vida se repite. Se habla de un ciclo de vida iterativo incremental porque el proceso se mejora en cada etapa.

El objetivo principal es desarrollar el modelo de proceso, que va a definir el flujo de proceso existentes en detalle. Se han modelado los procesos de negocio para satisfacer los siguientes objetivos:

- Para especificar el resultado exacto de los procesos de negocio, y para comprender el valor de negocio de este resultado.

- Para entender las actividades del proceso de negocio.

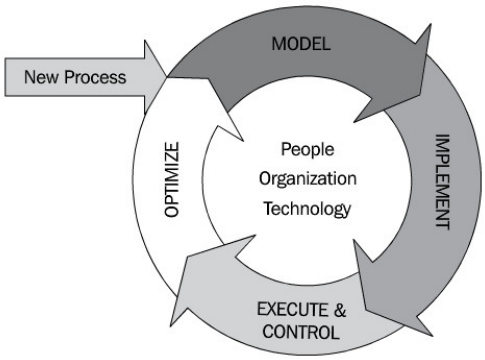


Figura 4.2.: Ciclo de vida de procesos de negocio [1]

- Para comprender el orden de las actividades.
- Para comprender las responsabilidades, identificar (y posteriormente supervisar) quien es responsable de las actividades y tareas.
- Para comprender la utilización de los recursos que se consumen en el proceso de negocio.
- Para entender la relación entre las personas involucradas en los procesos, y su comunicación.
- Para entender el flujo de documentos. Los procesos de negocio que producen y consumen los documentos (independientemente de si se trata de documentos en papel o electrónicos). Entender cuando los documentos se van, y donde están viniendo es importante.
- Para identificar los cuellos de botella y puntos potenciales de mejoras, que pueden ser utilizados más tarde en la fase de optimización de procesos.
- Para introducir los estándares de calidad como la International Standards Organization (ISO) 9001 con más éxito, y para pasar mejor certificación.
- Para mejorar la comprensibilidad de las normas de calidad que se pueden complementar con los diagramas de proceso.
- Para utilizar los modelos de procesos de negocio como líneas de trabajo para los nuevos empleados que presenten a los procesos de negocio más rápido y más eficientemente.
- Para entender los procesos de negocio, lo que permitirá comprender y describir la sociedad en su conjunto.

Una buena comprensión de los procesos de negocio es muy importante para el desarrollo de soporte de IT. Las aplicaciones que proporcionan soporte de extremo a extremo para los procesos de negocio, se pueden desarrollar de manera eficiente sólo si se entienden los procesos de negocio detalladamente.

BPMN [13] está diseñado para cubrir muchos tipos de modelos y permite la creación de procesos de negocio de extremo a extremo. Los elementos estructurales de BPMN permiten que el espectador sea capaz de diferenciar fácilmente entre las secciones de un diagrama de BPMN.

Para el diseño del proceso lo primero es comprender el resultado de negocio. Cuando ha sido identificado, se debe entender el flujo del proceso. Este se compone de actividades (o tareas) que se realizan en un orden determinado. El flujo de proceso se modela en diversos niveles de abstracción. En el más alto nivel de abstracción, el flujo del proceso muestra sólo las actividades más importantes. Cada una de las actividades del nivel superior de abstracción se descomponen en flujos detallados. La complejidad del proceso, y el nivel de

detalle requerido, son los criterios que indican como se debe descomponer. Para entender el comportamiento del proceso completo, tiene sentido descomponer hasta que se alcanzan actividades atómicas (es decir, actividades que no se pueden descomponer aún más).

En el desarrollo del modelo de proceso, una de las cosas más importantes a considerar es el nivel de detalle. Con el fin de proporcionar apoyo de extremo a extremo de los procesos de negocio utilizando SOA, debe hacerse el modelado de proceso detallado. En el diseño del proceso, se debe entender la estructura detallada de los procesos de negocio. Por lo tanto, se debe identificar por lo menos lo siguiente:

- Actividades de proceso en varios niveles de detalle.
- Roles responsables de la realización de cada actividad del proceso.
- Los eventos que desencadenan la ejecución de procesos y eventos que interrumpen el flujo del proceso.
- Los documentos intercambiados dentro del proceso. Esto incluye documentos de entrada y de salida.
- Las reglas de negocio que forman parte del proceso.

El enfoque habitual para el diseño del proceso incluye los siguientes pasos:

- La identificación de los roles.
- La identificación de las actividades.
- Conexión de las actividades a los roles.
- Definir el orden de las actividades.
- Adición de eventos.
- Adición de documentos.

También, se debe entender la eficiencia del proceso de negocio. Esto incluye la utilización de los recursos, el tiempo empleado por los empleados involucrados, los posibles cuellos de botella e ineficiencias. Esta es la razón por la que también debe identificar las métricas que se utilizan para medir la eficiencia del proceso. Se debe identificar si el proceso cumple con los estándares o procesos de referencia. También, hay que identificar los eventos que pueden interrumpir el flujo del proceso.

La orientación SOA permite modelar un proceso como una "orquestación" de servicios. En la terminología SOA componer nuevos servicios basándonos en los ya existentes se llama "orquestación".

4.1.2. Arquitectura orientada a servicios

SOA [2] (Arquitectura Orientada a Servicios) establece un modelo de arquitectura que tiene como objetivo mejorar la eficiencia, agilidad y productividad de una empresa posicionando los servicios como un mecanismo esencial para el logro de las metas estratégicas de la organización. Una implementación de SOA puede consistir en una combinación de tecnologías, productos, API, soportando extensiones de infraestructura, y algunas otras partes. La implementación de la arquitectura orientada a servicios es única dentro de cada empresa, sin embargo, se caracteriza por la introducción de nuevas tecnologías y plataformas que apoyan de manera específica a la creación, ejecución y evolución de las soluciones orientada a servicios.

Es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio. Son una serie de servicios reutilizables y recombinables independientemente para definir nuevos procesos de negocio y/o refinar los existentes. El concepto de "servicio" en SOA se basa en que los servicios desarrollados son "Servicios de Negocio", sean desarrollados mediante una tecnología

como “servicios web” o bien en otras tecnologías disponibles para ello. En una arquitectura SOA, el foco de interés se pone en el modelado de procesos a partir de piezas reutilizables que configuran servicios de negocio, abstrayéndose de la tecnología y forma en la que esos servicios son implementados.

Los principios de implementación definen las características físicas que debería tener en cuenta para diseñar servicios. La arquitectura SOA se asienta sobre los siguientes principios:

1. **Contratos de servicio estandarizados:** Por medio de un contrato los servicios expresan su funcionalidad. Las interfaces de entrada y salida tienen que estar declaradas explícitamente, utilizando por ejemplo WSDL. El consumidor del servicio accederá a este mediante el contrato, logrando así la independencia entre el consumidor y la implementación del propio servicio.

2. **Servicios con bajo acoplamiento:** Independencia entre servicios.

3. **Abstracción:** El contrato de un servicio contiene la información esencial, los servicios ocultan la lógica a los demás.

4. **Reusabilidad:** Reaprovechamiento de los servicios encapsulados para ser utilizados por otros servicios.

5. **Autonomía:** Los servicios tienen control sobre la lógica que encapsulan, desde una perspectiva de diseño y ejecución.

6. **Sin estado:** Los servicios minimizan el consumo de recursos aplazando la gestión de la información de estado cuando sea necesario.

7. **Capacidad de descubrimiento:** Gracias a los metadatos, el servicio puede ser descubierto de forma eficaz, utilizando para ello registros de servicios como Universal Description, Discovery and Integration (UDDI).

8. **Composición:** Los servicios de más bajo nivel componen servicios complejos de más alto nivel.

9. **Interoperabilidad:** Entre servicios que se ejecutan en diferentes entornos y sistemas operativos.

El principal objetivo de SOA a través de los servicios es construir software altamente reutilizable para lo cual se basa en un conjunto de técnicas y principios. Cuando se construyen diversos tipos de servicios, es evidente que se pueden clasificar en función del tipo de lógica que encapsulan, del alcance de potencial de reutilización que tiene esta lógica y de cómo esta lógica se relaciona con dominios existentes dentro de la empresa. Como resultado, hay tres categorías comunes que representan los principales modelos de servicios:

- Servicios de entidad.

- Servicios de tarea.

- Servicios de utilidad.

El uso de estos modelos de servicio da lugar a la creación de capas de abstracción lógica de servicio, como se muestra en la figura 4.3. Los modelos de servicio establecen capas de abstracción de servicios comunes, un servicio de nivel inferior puede formar parte de servicios de nivel superior o procesos de negocio, posiblemente colaborando con otros servicios, y con un cierto nivel de autonomía.

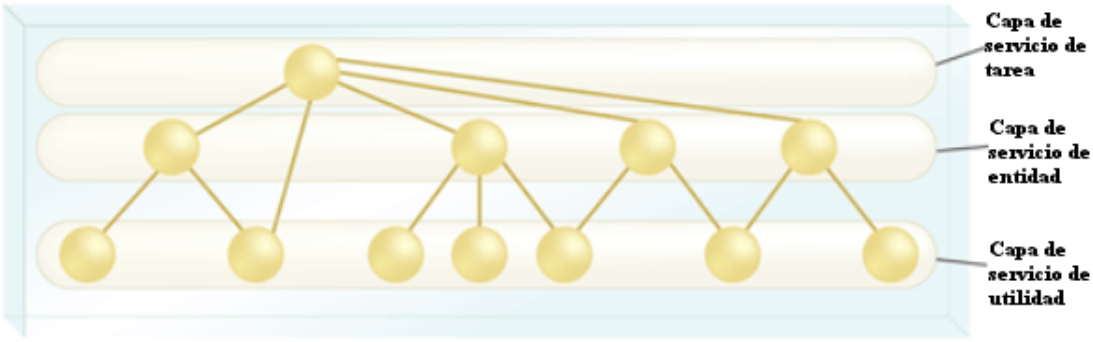


Figura 4.3.: Capas de servicios [2].

Servicios de entidad

Son aquellos que están centrados en el contexto de las entidades de negocio. Algunos ejemplos de entidades de negocios son: los clientes, los empleados y la factura. Es considerado un servicio altamente reutilizable porque es agnóstico (son independientes de donde se ejecutan y en que proceso lo hacen) a la mayoría de los procesos de negocio padre. Por consiguiente, un único servicio de entidad se puede aprovechar para automatizar múltiples procesos de negocio padre. Muchas de las operaciones que exponen este tipo de servicios son las típicas de un CRUD. Evidentemente, estos servicios variarán en función del negocio concreto y las entidades que representen su actividad.

Servicios de tarea

Son aquellos que engloban un proceso de negocio apoyándose (normalmente) en servicios de utilidad o de entidad. Este tipo de servicio tiende a tener menos potencial de reutilización y suelen consistir en una serie de pasos para completar una tarea específica. Al surgir como respuesta a una necesidad concreta del negocio, su funcionalidad variará en función de los cambios del propio negocio. Es muy importante tener una buena base de servicios de entidad y utilidad sobre los que se apoyen los servicios de tarea para poder responder al cambio de manera ágil (composición de servicios). Un ejemplo podría ser un servicio de autorización de prestación de servicio. Se supone el caso en que un cliente de una aseguradora va al médico a pasar consulta. Se podría tener un servicio que validase que el cliente puede ir a ese médico a pasar consulta. Para realizar la validación probablemente el servicio debería apoyarse en diferentes servicios de entidad, concretamente en las entidades de negocio que intervienen en el proceso y, probablemente, en algún servicio de utilidad (por ejemplo, enviar un correo con el resultado de la validación).

Servicios de utilidad

Cada uno de los modelos de servicio descritos anteriormente tiene un enfoque muy claro sobre lo que representa la lógica de negocio. El modelo de servicio de utilidad no está relacionado con el negocio. Se dedica a proporcionar funcionalidad de utilidad transversal, reusabilidad, tales como el registro de eventos, notificación y manejo de excepciones. Son servicios que no cubren una necesidad concreta de negocio. Estos servicios contienen un alto potencial de reusabilidad. Algunos ejemplos: un servicio de gestión de «tokens» de seguridad de acceso a aplicaciones o servicios de la plataforma, un servicio de envío de correos.

4.2. Diseño de la base de datos

Todo sistema informático que maneje una gran cantidad de datos es imprescindible que utilice una base de datos. Una base de datos es una colección de datos relacionados, donde los datos son hechos grabados, que están relacionados entre sí y que tienen un significado implícito [40]. Tiene las siguientes propiedades implícitas:

- Representa algún aspecto del mundo real.
- Es una colección de datos almacenados lógicamente coherentes con algún tipo de significado incoherente.
- Una base de datos se diseña, construye y rellena con datos para un propósito específico.

Se van a realizar las siguientes fases de diseño, las cuales se explicarán durante este capítulo:

- Diseño Conceptual.
- Diseño Lógico.

4.2.1. Diseño conceptual

El diseño conceptual [41] de una base de datos proporciona una visión detallada del desarrollo. Se ha usado el modelo Entidad-Relación (ER) para desarrollar el esquema conceptual. Especifica todos los conjuntos de entidades, conjuntos de relaciones, atributos y restricciones de correspondencia. El diseñador revisa el esquema para confirmar que todos los requisitos de datos se satisfacen realmente y no hay conflictos entre sí. También, se examina el diseño para eliminar características redundantes. Lo importante en este punto es describir los datos y las relaciones, más que especificar detalles del almacenamiento físico. En esta fase de diseño conceptual se puede hacer una revisión del esquema para encontrar los requisitos funcionales.

Recogida de datos

El sistema debe de recoger toda la información necesaria de los usuarios registrados en el sistema. Estos usuarios son las oficinas del registro de la propiedad, los notarios y las entidades de crédito que pueden ser los bancos, los seguros y los fondos de pensiones. Los propietarios no tendrán acceso, se seguirá visitando a la entidad de crédito para las solicitudes de crédito y al notario para los asuntos relativos al registro de la propiedad.

El estar registrado, permite tener acceso al sistema y poder realizar ciertas gestiones de forma manual. Los usuarios obligatoriamente deben pertenecer a uno o varios perfiles como son: administrador del sistema, entidad de crédito, registro de la propiedad y notario.

Al poseer la documentación suiza, tras traducirla se recogió todos los datos necesarios. La mayor parte de los datos que interesan y que deben ser almacenados en la base de datos son los que forman parte de los formularios que los usuarios deben rellenar.

Restricciones semánticas

La primera parte del diseño conceptual es la recogida de datos; y una vez que se han recogido los datos, es necesario determinar aquellos de los que va a constar el diseño conceptual. Los tipos de datos que han sido necesarios son:

- Las diferentes entidades donde se van a recoger toda la información necesaria.
- Los atributos de cada una de las entidades y los tipos de estos.

- Las relaciones necesarias para unir dos o más entidades y así poder conformar la estructura de la base de datos.
- Las restricciones a tener en cuenta a la hora del diseño de la base de datos.

Tipos de entidades

Se va a pasar a determinar los tipos de entidades que han sido necesarios para el correcto diseño e implementación de la base de datos, en la tabla 4.1.

Además, se va a dar una breve explicación de cada tipo de entidad que intervienen, indicando su tipo, si es fuerte o débil, los atributos que posee y su identificador.

Antes de mostrar la tabla, es necesario destacar la diferencia entre un tipo de entidad fuerte o débil. Los tipos de entidad fuertes son aquellos tipos de entidad regulares que tienen un atributo clave, mientras que, los tipos de entidad débiles son aquellos tipos de entidad que no tienen atributos clave propios [40].

Entidades	**Descripción**	**Tipo**	**Atributos**	**Identificador**
Entidad	Clase padre creada para poder diferenciar el remitente y el destinatario del formulario. Existen 3 tipos de entidades: entidad de crédito, registro de la propiedad y notario	Fuerte	numPart, tipo, IDE, direccion, codPostal, lugar	numPart
Entidad_de_Credito	Tipo de entidad. Es la denominación de cualquier empresa cuya finalidad o actividad es la de otorgar créditos a terceros. Son los bancos, seguros y fondos de pensiones (Acreedores)	Débil	empresa, formaJuridica, sede	numPart
Registro_Propiedad	Tipo de Entidad. Es el catastro inmobiliario. Es una institución que se ocupa de la inscripción del dominio y demás derechos reales sobre bienes inmuebles	Débil	oficinaCatastro	numPart
Notario	Tipo de Entidad. Su intervención otorga carácter público a los documentos privados, autorizándolos a tal fin con su firma. En Suiza, el notariado está sujeto mayoritariamente a la legislación cantonal. Tres tipos de notarios: independiente (AG, BE, BS, FR, GE, JU, NE, TI, UR, VD, VS), administrativo (AR, SH, TG, ZH) y de forma mixta (AI, BL, GL, GR, LU, NW, OW, SG, SO, SZ, ZG)	Débil	codActivacion, nombre, apellidos, notarioEnCanton, regNotariosCanton	numPart

Entidades	Descripción	Tipo	Atributos	Identificador
Transaccion	Es una interacción con una estructura de datos compleja, compuesta por varios procesos que se han de aplicar uno después del otro. La transacción debe realizarse de una sola vez. Se intercambia notificaciones electrónicas entre las oficinas del registro de la propiedad, los notarios, las entidades de crédito	Fuerte	numTransac, tipoNot, fechaHora	numTransac
Hipoteca	Es ante todo un derecho real de garantía y de realización de valor, que se constituye para asegurar el cumplimiento de una obligación (normalmente de pago de un crédito o préstamo) sobre un bien, (generalmente inmueble), nace de un contrato, de modo que en el instante inicial, antes de su inscripción en el registro de la propiedad, con la cual nace y adquiere la condición de derecho real eficaz frente a terceros, la hipoteca es un contrato. Crédito garantizado por una garantía inmobiliaria, que se concede por las entidades de crédito	Fuerte	importe, caseHypothecaire, tipo, fechado, numDoc, EREID, tasaIntMax, paridadRango, derechoAscRangoAnt	EREID
Inmueble	Cada una de las propiedades que se van a registrar y que forman parte de la hipoteca. Es un espacio delimitado en el suelo.	Fuerte	municipioBarrio, denominacion, numCatastro, hojaReg, numInmueble, EGRID	numInmueble
Sujeto	Quien quiere hacer la hipoteca para registrar inmuebles a su nombre (Propietario - Deudor)	Fuerte	IDE_NIP, tipo	IDE_NIP
Comunidad	Tipo de sujeto. Conjunto de personas físicas que elaboran, comparten y socializan una identidad común, diferenciándose de otras comunidades	Débil	nombre, fchConstitucion	IDE_NIP
Persona_Juridica	Tipo de sujeto. Entidad que tiene ciertos derechos, pero también ciertas obligaciones	Débil	empresa, formaJuridica, sede, direccion, codPostal	IDE_NIP

Entidades	Descripción	Tipo	Atributos	Identificador
Persona_Fisica	Tipo de sujeto. Las personas físicas son los seres humanos. Toda persona física tiene los derechos que la Constitución y las demás normas le otorgan. También, es cada una de las personas que forman una comunidad	Débil	nombre, apellidos, fchNacimiento, lugarOrigen, nacionalidad, sexo, estadoCivil, dirección, codPostal, lugar	IDE_NIP
Referencia	Las distintas referencias que tiene la hipoteca	Fuerte	refInterna	refInterna
Usuario	Miembros registrados	Fuerte	nombre, apellidos, telefono, email, numUsuario, username, password	numUsuario
Rol	Diferentes perfiles de los usuarios	Fuerte	id, rol	id
Documento	Documentos registrados	Fuerte	numDoc, nombrePDF, fchconst, link	numDoc
Promesa_Pago	En la adquisición de hipotecas entre entidades de crédito se realiza una promesa de pago irrevocable, condicional y limitada en el tiempo. Acuerdo entre las entidades de crédito en el pago del cambio de acreedor. La presente promesa de pago es irrevocable hasta la fecha de validez. Expira automáticamente y totalmente a esta fecha si las condiciones no son cumplidas hasta entonces	Fuerte	id, condiciones, validoHasta	id
Plazo	Cuotas en las que se lleva a cabo la promesa de pago. Precisiones sobre el pago	Fuerte	numPlazo, moneda, importe, fechaValida, beneficiario, iban	numPlazo

Tabla 4.1.: Tabla de Entidades.

Tipos de relaciones

Se va a pasar a determinar las diferentes relaciones que se han visto necesarias para relacionar las distintas entidades determinadas en el apartado anterior en la tabla 4.2. Además, de determinar las diferentes relaciones se ha escrito una breve descripción de las mismas, las entidades que unen y la cardinalidad que poseen. También, se ha determinado si posee o no atributos propios de la relación y cuáles son.

Relaciones	Descripción	Entidades	Atributos
concede	Una entidad de crédito concede una o muchas hipotecas mediante una o ninguna promesa de pago	Entidad_de_Credito, Hipoteca y Promesa_Pago; Cardinalidad: M:N:1	fchIni, fchFin, fchFirma
participa_en	Un sujeto participa en diferentes hipotecas	Sujeto e Hipoteca; Cardinalidad: M:N	rol, fchIni, fchFin, fchFirma
se_relaciona_con	Cada una de las entidades de crédito contacta con un registro de la propiedad	Entidad_de_Credito y Registro_Propiedad; Cardinalidad: 1:N	
realiza	El registro de la propiedad realiza una o más transacciones	Entidad y Transaccion; Cardinalidad: 1:N	
registra	Cuando se realiza una transacción se registra una o muchas hipotecas	Transaccion e Hipoteca; Cardinalidad: M:N	
contiene	La hipoteca contiene uno o más inmuebles	Hipoteca e Inmueble; Cardinalidad: 1:N	
tiene	La hipoteca tiene una o más referencias	Referencia e Hipoteca; Cardinalidad: N:1	
formado_por	La comunidad esta formada por una o más personas físicas	Comunidad y Persona_Fisica; Cardinalidad: 1:N	
guarda	Una transacción guarda ningún, uno o muchos documentos	Transaccion y Documento; Cardinalidad: 1:N	
se_registra_con	El usuario se registra con diferentes roles	Usuario y Rol; Cardinalidad: M:N	
seguido_por	En referencia el proceso es seguido por un usuario	Referencia y Usuario; Cardinalidad: 1:N	
fimante	En referencia el firmantes es ningún, uno o dos usuarios	Referencia y Usuario; Cardinalidad: 1:N	funcion
compuesta_de	Una promesa de pago esta compuesta por uno o muchos plazos	Promesa_Pago y Plazo; Cardinalidad: 1:N	

Tabla 4.2.: Tabla de Relaciones.

Restricciones

A la hora de diseñar una base de datos es necesario tener en cuenta una serie de restricciones con el fin de representar de la forma más precisa la realidad con la que se está trabajando.

1. Una entidad de crédito concede una o muchas hipotecas y una hipoteca puede ser concedida por una o muchas entidades de crédito (Cambios de acreedor). Se concede con una una o ninguna promesa de pago (Acreedor comprador y vendedor).

2. Una promesa de pago está compuesta por uno o muchos plazos y uno o dichos plazos componen una única promesa de pago.

3. Un sujeto participa en una o muchas hipotecas mientras que en una hipoteca pueden participar varios sujetos (Propietario - Deudor).

4. Un único registro de la propiedad para muchas entidades de crédito del mismo distrito y muchas entidades de crédito del mismo distrito se relacionan con un único registro de la propiedad.

5. Una o muchas transacciones son realizadas por una entidad y una entidad realiza muchas transacciones.

6. Cuando se realiza una transacción se registra una o muchas hipotecas y una hipoteca varios inmuebles y todos los inmuebles conciernen al mismo distrito del registro de la propiedad.

7. Un inmueble va a pertenecer a una única hipoteca.

8. Una comunidad está formada por una o varias personas físicas.

9. Una hipoteca tiene una o varias referencias, cada una de las referencias están seguidas por un usuario y ningún, uno o dos firmantes.

10. Un mismo usuario puede estar registrado en el sistema con más de un rol.

11. Un mismo rol puede pertenecer a varios usuarios.

12. Una transacción guarda ningún, uno o muchos documentos y un documento es guardado por una única transacción.

Descripción de atributos

En tabla 4.3, se describe cada uno de los atributos necesarios en la base de datos. Para cada uno de los atributos se va a decir la entidad o relación a la que pertenece, el nombre que posee en la base de datos, una breve descripción, el tipo de atributo y de datos que posee, además, del dominio, se indica si puede tomar o no valores nulos y si es o no clave primaria.

Se ha de resaltar un conjunto de atributos que son de tipo compuesto. Los atributos compuestos se manejan creando un atributo separado para cada uno de los atributos componentes; no se crea una columna separada para el propio atributo compuesto [41]. En esta tabla no se define dicho conjunto de atributos por lo que se ha visto adecuado nombrarlos:

- Fechas: Todas las fechas van a estar compuestas de día, mes y año.
- Direcciones: Todas las direcciones pueden estar compuestas de vía, número, bloque, escalera y puerta.
- Apellidos: Todos los apellidos pueden estar compuestos de primer y segundo apellido o sólo el primero.
- Lugares: Todos los lugares van a estar compuestos de municipio, distrito, cantón.
- MunicipioBarrio: Este atributo va a estar compuesto de barrio, municipio y distrito al que pertenece el inmueble.
- Lugar de origen: El lugar de origen va a estar compuesto de municipio, distrito, cantón.

También, se ha de destacar la diferencia entre un atributo multivaluado y univaluado. Un atributo multivaluado puede tomar más de un valor a la vez y un atributo univaluado toma un solo valor. En este caso una persona puede tener más de un teléfono.

Finalmente, hay ciertos atributos a resaltar porque siguen siempre una misma estructura, como son:

- Código postal: Consta de 4 dígitos y cada uno da una indicación geográfica del lugar a las que corresponden. Primer dígito, gran área geográfica; segundo dígito, área más pequeña entorno a una ciudad; tercer dígito, trayecto; cuarto dígito, ubicación.
- International Bank Account Number (IBAN): Código del país + código de control + identificador del banco + número de cuenta. El IBAN esta compuesto de 2 letras que son el código del país, 2 números de control y hasta un máximo de 30 posiciones más para la cuenta.
- NIP: Tiene 13 cifras, un prefijo de 3 dígitos que identifican el país emisor, 9 dígitos de numeración que forman la parte identificándose y un dígito de control.
- IDE: Consta del código del país y 9 números siendo el último la cifra de control, calculada a través del «standard» módulo 11.
- EREID: Consta de 14 caracteres alfanuméricos. Estructura: CH + 12 dígitos del 0 al 9, siendo los dos últimos dígitos de control.
- EGRID: Consta hasta de 22 caracteres alfanuméricos. Estructura: CH + prefijo (4 dígitos) + ID (14 dígitos de conjunto de caracteres restringido) + dígito de control (2 dígitos).

Diagrama de Entidad-Relación (DER)

El DER de este proyecto se ha dividido en diferentes figuras porque es demasiado extenso y se va a mostrar mejor de forma detallada. Como se puede observar en la figura 4.4, se muestra la parte genérica de este diagrama porque Hipoteca es la que se relaciona con un mayor número de entidades. En la figura 4.5 se muestra la entidad Promesa_Pago y en la figura 4.6, la entidad Sujeto, pudiendo ser, comunidades, personas físicas y personas jurídicas. En la figura 4.7 se muestra las relaciones con la entidad Transaccion y en la figura 4.8, las relaciones con la entidad Referencia. Finalmente, en la figura 4.9 se muestra las relaciones con Entidad, pudiendo ser, entidades de crédito, registros de la propiedad y notarios.

Entidad							
Entidad							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Número de participantes	numPart	Identificación de los participantes registrados	Simple	Numérico	Natural	No nulo	numPart
Tipo	tipo	Tipo de entidad (registro de la propiedad, entidad de crédito o notario)	Simple	Carácter	Alfabético	No nulo	
IDE	IDE	Número de identificación de empresa de la entidad	Simple	Carácter y numéricos	Alfabético y natural	No nulo	
Dirección	direccion	Dirección de la entidad	Compuesto	Carácter y numéricos	Alfabético y natural	No nulo	
Código Postal	codPostal	Código postal de la entidad	Simple	Numérico	Natural	No nulo	
Lugar	lugar	Lugar en el que se encuentra la entidad	Compuesto	Carácter	Alfabético	No nulo	
Entidad							
Entidad_de_Credito							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Empresa	empresa	Nombre de la empresa de la entidad de crédito	Simple	Carácter	Alfabético	No nulo	
Forma juridica	formaJuridica	Es uno de los elementos de su estatus. Posibles formas jurídicas: la empresa individual, sociedad colectiva, sociedad de responsabilidad limitada, la sociedad anónima y sociedad en comandita (sociedad comercial de capital)	Simple	Carácter	Alfabético	No nulo	
Sede	sede	Lugar donde tiene su domicilio la entidad de crédito	Simple	Carácter	Alfabético	No nulo	

Entidad							
Registro_Propiedad							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Oficina del catastro	oficinaCatastro	Nombre de la oficina del catastro	Simple	Carácter	Alfabético	No nulo	
Entidad							
Notario							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Código de activación	codActivacion	Código de activación que la entidad de crédito le da al sujeto y el sujeto al notario	Simple	Numérico	Natural	No nulo	
Nombre	nombre	Nombre del notario	Simple	Carácter	Alfabético	No nulo	
Apellidos	apellidos	Apellidos del notario	Compuesto	Carácter	Alfabético	No nulo	
Notario en cantones	notarioEnCanton	Cantones en los que trabaja dependiendo del tipo de notario que sea	Compuesto	Carácter	Alfabético	No nulo	
Registro notarios cantón	registroNotariosCanton	Número del registro en el registro de notarios	Simple	Numérico	Natural	No nulo	
Entidad							
Transaccion							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Número de transacción	numTransac	Número de transacción otorgado por el sistema	Simple	Numérico	Natural	No nulo	numTransac
Tipo de notificación	tipoNot	Breve descripción de lo que se va a realizar en la transacción	Simple	Carácter	Alfabético	No nulo	
Fecha transacción	fechaHora	Fecha y hora en la que se realiza la transacción	Compuesto	Numérico	Natural	No nulo	

Entidad							
Hipoteca							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Importe	importe	Importe de la hipoteca	Simple	Numérico	real	No nulo	
Case hypothecaire	caseHypothecaire	Identificador del case hypothecaire	Simple	Numérico	Natural	No nulo	
Tipo de hipoteca	tipo	Tipos de cédula hipotecaria, puede ser de registro	Simple	Carácter	Alfabético	No nulo	
Fecha constitución	fechado	Fecha en la que la hipoteca fue constituida	Compuesto	Numérico	Natural	No nulo	
Número de documento	numDoc	Número del documento de la hipoteca	Simple	Numérico	Natural	No nulo	
EREID	EREID	Código de identificación inequívoca de los derechos en el registro de la propiedad. ID electrónico de los derechos	Simple	Carácter y numérico	Alfabético y natural	No nulo	EREID
Tasa de interés máxima	tasaIntMax	Es un monto de dinero que se traduce en un porcentaje y que el deudor deberá pagar a quien le presta, por el uso de ese dinero	Simple	Numérico	Natural	Nulo	
Paridad de rango	paridadRango	Cláusula que permite que hipotecas posteriores a la primera y a favor del mismo acreedor, a la hora de ejecutar, se sumen y se reclame una deuda mayor	Simple	Carácter	Alfabético	Nulo	
Derecho de ascenso a un rango anterior	derechoAscRangoAnt	Con o sin derecho de ascenso a un rango anterior	Simple	Carácter	Alfabético	Nulo	

Entidad							
Inmueble							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Municipio y Barrio	municipioBarrio	Municipio y barrio donde se encuentra el inmueble	Compuesto	Carácter	Alfabético	No nulo	
Denominación	denominacion	Denominación del inmueble. Breve descripción sobre las características del inmueble	Simple	Carácter	Alfabético	No nulo	
Referencia Catastral	numCatastro	Número del catastro	Simple	Carácter y numérico	Alfabético y natural	No nulo	
Hoja del registro de la propiedad	hojaReg	Hoja del registro de la propiedad	Simple	Numérico	Natural	No nulo	
Número del inmueble	numInmueble	Número del inmueble	Simple	Numérico	Natural	No nulo	numInmueble
EGRID	EGRID	Código de identificación. ID electrónico de la parcela	Simple	Carácter y numérico	Alfabético y natural	No nulo	
Entidad							
Sujeto							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
IDE o NIP	IDE_NIP	IDE o NIP del tipo de sujeto. NIP de la persona física o NIP de la comunidad o IDE de la persona jurídica. Número de identificación personal único, a todas las personas en Suiza se le asigna un número de la seguridad social al nacer o cuando se establecen por primera vez en Suiza	Simple	Carácter y numérico	Alfabético y natural	No nulo	IDE_NIP
Tipo de sujeto	tipo	Tipo de sujeto	Simple	Carácter	Alfabético	No nulo	

Entidad							
Comunidad							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Nombre	nombre	Nombre de la Comunidad	Simple	Carácter	Alfabético	No nulo	
Fecha constitución	fchConstitucion	Fecha en la que se creó la comunidad	Compuesto	Numérico	Natural	No nulo	
Entidad							
Persona_Juridica							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Empresa	empresa	Nombre de la empresa	Simple	Carácter	Alfabético	No nulo	
Forma jurídica	formaJuridica	Es uno de los elementos de su estatus. Posibles formas jurídicas: la empresa individual, sociedad colectiva, sociedad de responsabilidad limitada, la sociedad anónima y sociedad en comandita (sociedad comercial de capital)	Simple	Carácter	Alfabético	No nulo	
Sede	sede	Lugar donde tiene su domicilio la persona jurídica	Simple	Carácter	Alfabético	No nulo	
Dirección	direccion	Dirección de la persona jurídica	Compuesto	Carácter y numérico	Alfabético y natural	No nulo	
Código Postal	codPostal	Código postal de la persona jurídica	Simple	Numérico	Natural	No nulo	

Entidad							
Persona_Fisica							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Nombre	nombre	Nombre de la persona física	Simple	Carácter	Alfabético	No nulo	
Apellidos	apellidos	Apellidos de la persona física	Compuesto	Carácter	Alfabético	No nulo	
Fecha de nacimiento	fchNacimiento	Fecha de nacimiento de la persona física	Compuesto	Numérico	Natural	No nulo	
Lugar de origen	lugarOrigen	Lugar de nacimiento de la persona física	Compuesto	Carácter	Alfabético	No nulo	
Nacionalidad	nacionalidad	Nacionalidad de la persona física	Simple	Carácter	Alfabético	No nulo	
Sexo	sexo	Sexo de la persona física	Simple	Carácter	Alfabético	No nulo	
Estado civil	estadoCivil	Estado civil de la persona física: casado(a), Soltero(a) y pareja de hecho registrada (contrato de unión civil exclusivamente para parejas homosexuales y que existe en Suiza desde el año 2007)	Simple	Carácter	Alfabético	No nulo	
Dirección	direccion	Dirección de la persona física	Compuesto	Carácter y numérico	Alfabético y natural	No nulo	
Código postal	codPostal	Código Postal de la persona física	Simple	Numérico	Natural	No nulo	
Lugar	lugar	Lugar en que reside la persona física	Compuesto	Carácter	Alfabético	No nulo	

Entidad							
Referencia							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Referencia interna	refInterna	Referencia interna que tiene la hipoteca	Simple	Carácter y numérico	Alfabético y natural	No nulo	refInterna
Entidad							
Usuario							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Nombre	nombre	Nombre del usuario	Simple	Carácter	Alfabético	No nulo	
Apellidos	apellidos	Apellidos del usuario	Compuesto	Carácter	Alfabético	No nulo	
Teléfono	telefono	Teléfono de contacto del usuario	Multivaluado	Numérico	Natural	No nulo	
Correo electrónico	email	Email de contacto del usuario	Monovaluado	Carácter y numérico	Alfabético y natural	No nulo	
Nombre del usuario	username	Nombre del usuario para el acceso al sistema	Simple	Carácter y numérico	Alfabético y natural	No nulo	
Clave	password	Clave del usuario para el acceso al sistema	Simple	Carácter y numérico	Alfabético, natural y signos de puntuación	No nulo	
Número del usuario	numUsuario	Número del usuario	Simple	Numérico	Natural	No nulo	numUsuario

Entidad							
Rol							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
ID	id	Código identificador del rol	Simple	numérico	natural	No nulo	id
Perfil	rol	Nombre del rol	Simple	Carácter	Alfabético	No nulo	
Entidad							
Documento							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Número del documento	numDoc	Número de identificación del documento	Simple	Numérico	Natural	No nulo	numDoc
Nombre de PDF	nombrePDF	Nombre del documento en PDF	Simple	Carácter y numérico	Alfabético, natural y signos de puntuación	No nulo	
Fecha de constitución	fchConstitucion	Fecha en la que se creó el documento	Compuesto	Numérico	Natural	No nulo	
Archivo «Link»	link	Elemento de un documento electrónico que permite acceder automáticamente a otro documento o a otra parte del mismo	Compuesto	Carácter y numéricos	Alfabético y natural	No nulo	
Entidad							
Promesa_Pago							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
ID	ID	Número de identificación de la promesa de pago	Simple	Numérico	Natural	No nulo	ID
Condiciones	condiciones	Condiciones de la promesa de pago	Compuesto	Carácter y numéricos	Alfabético y natural	No nulo	
Válido hasta	validoHasta	Fecha de validez de la promesa de pago	Compuesto	Numérico	Natural	No nulo	

Entidad							
Plazo							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Número del plazo	numPlazo	Número de identificación del plazo de la promesa de pago	Simple	Numérico	Natural	No nulo	numPazo
Moneda	moneda	Moneda con la que se va a realizar el importe	Simple	Carácter	Alfabético	No nulo	
Importe	importe	Importe del plazo de la promesa de pago	Simple	Numérico	real	No nulo	
Fecha válida	fechaValida	Fecha de validez del plazo	Compuesto	Numérico	Natural	No nulo	
Beneficiario/a	beneficiario	Destinatario del pago (IDE de la entidad de crédito destinataria)	Simple	Carácter y numéricos	Alfabético y natural	No nulo	
IBAN	iban	Identifica la misma cuenta bancaria. Puede utilizarse en todas las operaciones con el extranjero, con todos los países del mundo, y sirve principalmente para identificar la cuenta bancaria en los pagos internacionales. Su objetivo es facilitar la identificación homogénea de las cuentas bancarias a todos los países	Simple	Carácter y numéricos	Alfabético y natural	No nulo	

Relación							
firmante							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Función	funcion	Función del firmante	Simple	Carácter	Alfabético	No nulo	
Relación							
participa_en							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Rol	rol	Rol para indicar si el que participa en la hipoteca es el propietario	Simple	Carácter	Alfabético	No nulo	
Fecha inicio	fchIni	Fecha de inicio en la que el sujeto participa en la hipoteca	Compuesto	Numérico	Natural	No nulo	
Fecha fin	fchFin	Fecha de finalización en la que el sujeto participa en la hipoteca	Compuesto	Numérico	Natural	Nulo	
Fecha de la firma	fchFirma	Fecha de la firma de la hipoteca	Compuesto	Numérico	Natural	No nulo	
Relación							
concede							
Nombre	**Nombre atributo**	**Descripción**	**Tipo atributo**	**Tipo datos**	**Dominio**	**Nulo**	**Clave**
Fecha inicio	fchIni	Fecha de inicio en la que la entidad de crédito concede la hipoteca	Compuesto	Numérico	Natural	No nulo	
Fecha fin	fchFin	Fecha de finalización en la que la entidad de crédito concede la hipoteca	Compuesto	Numérico	Natural	Nulo	
Fecha de la firma	fchFirma	Fecha de la firma de la hipoteca	Compuesto	Numérico	Natural	No nulo	

Tabla 4.3.: Tabla de atributos.

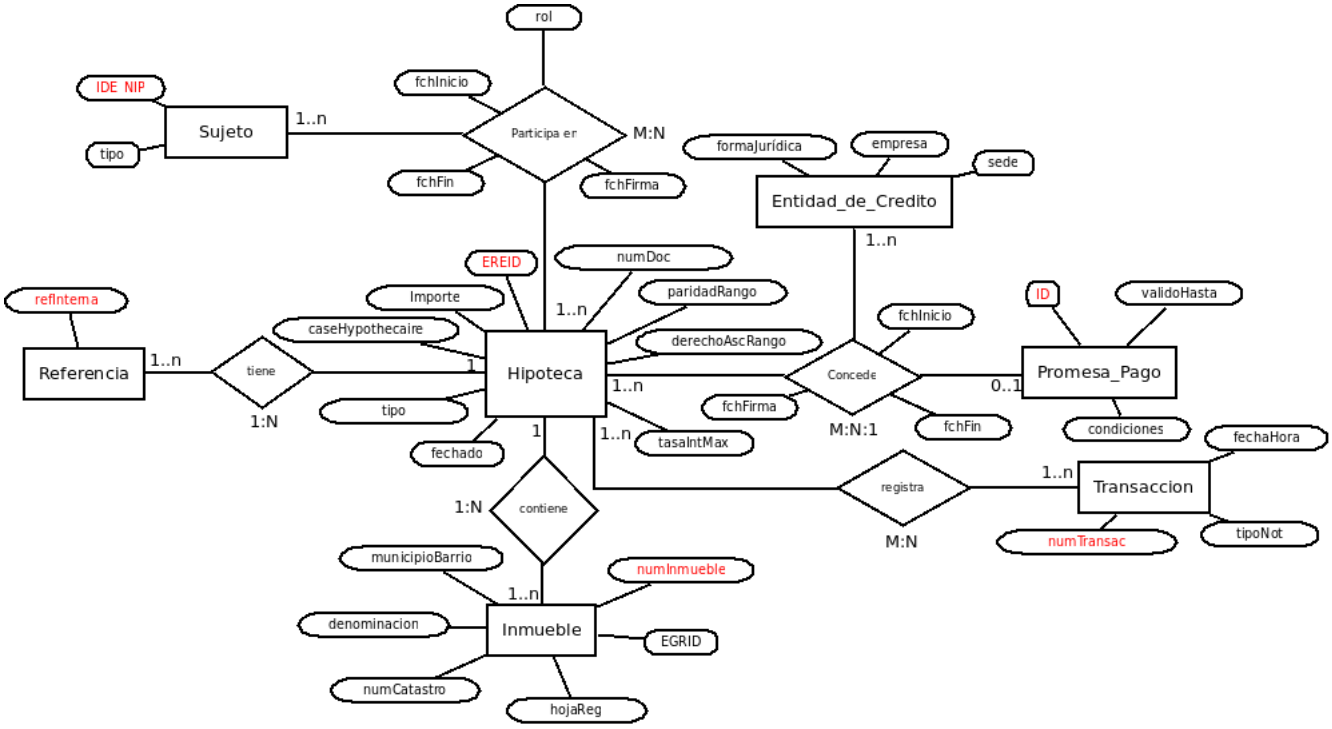


Figura 4.4.: Diagrama de Entidad-Relación (Hipoteca)

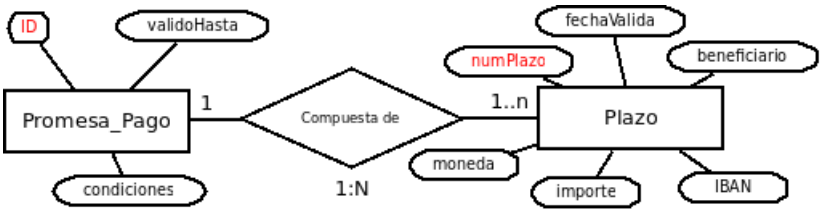


Figura 4.5.: Diagrama de Entidad-Relación (Promesa de pago)

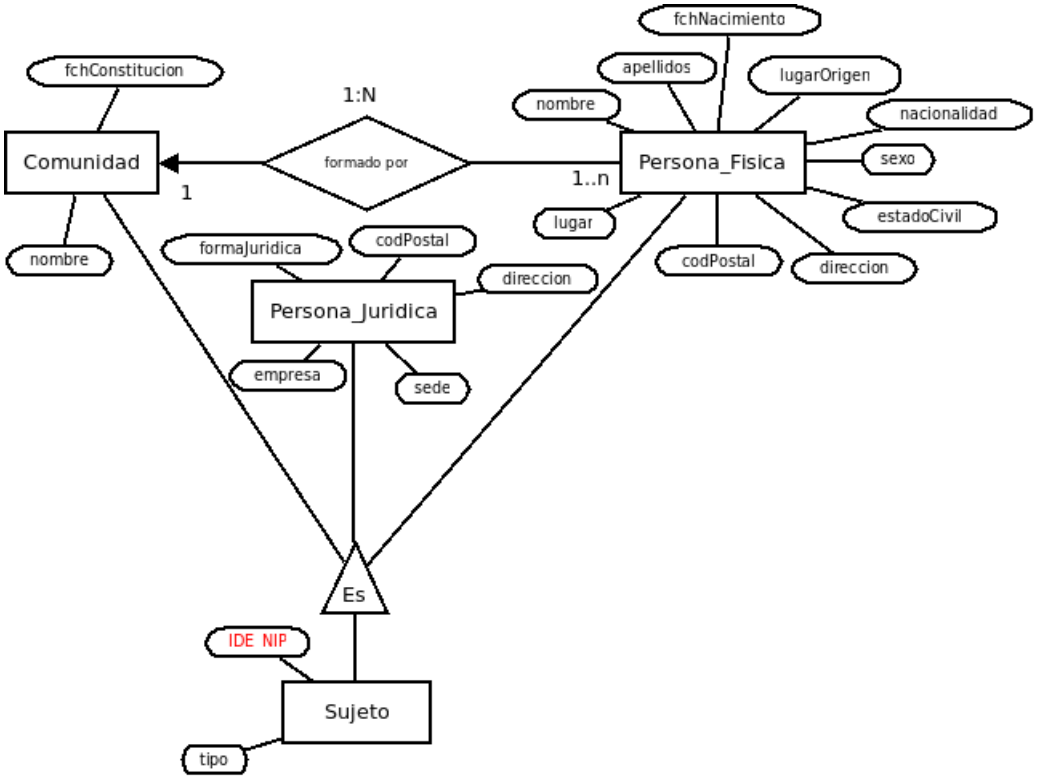


Figura 4.6.: Diagrama de Entidad-Relación (Sujeto)

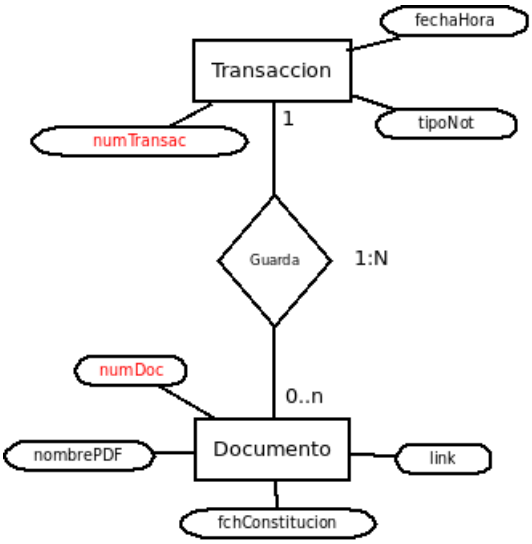


Figura 4.7.: Diagrama de Entidad-Relación (Transaccion)

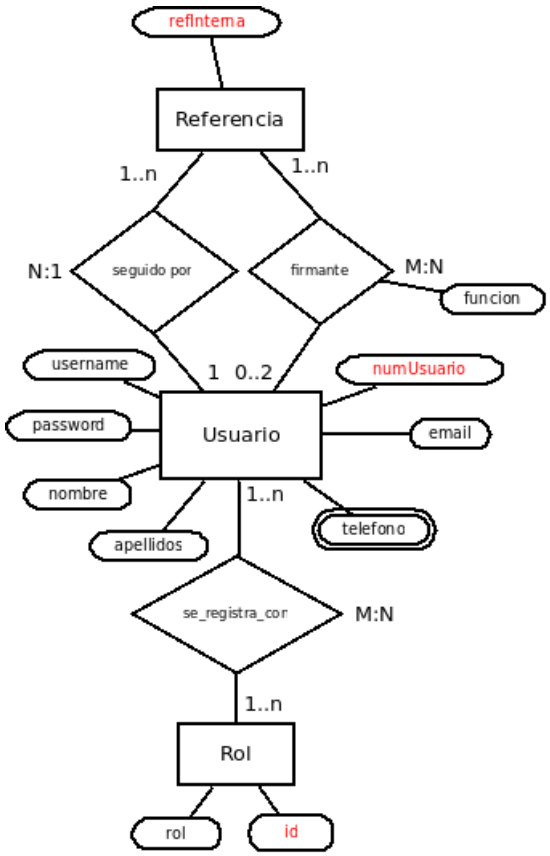


Figura 4.8.: Diagrama de Entidad-Relación (Referencia)

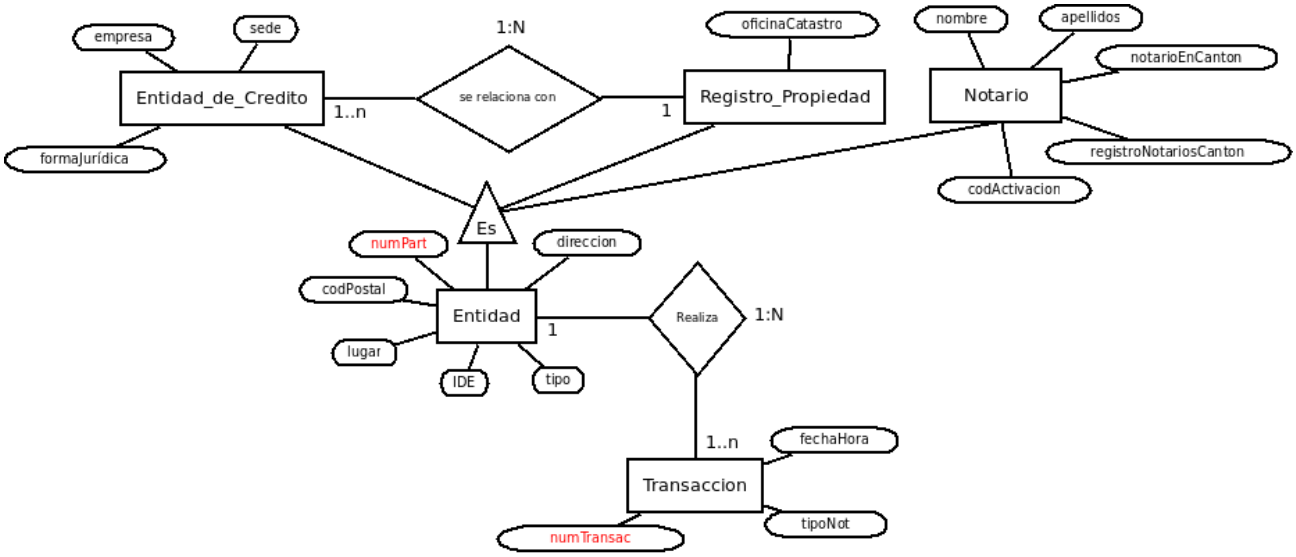


Figura 4.9.: Diagrama de Entidad-Relación (Entidad)

4.2.2. Diseño lógico

En el diseño lógico [41] se traduce el esquema conceptual de alto nivel anteriormente descrito al modelo de datos de la implementación del sistema de base de datos que se usará. No es más que una descripción de la estructura de la base de datos. Se va a describir como los elementos en la base de datos tendrán que quedar agrupados. Además, durante esta parte del diseño la representación debe de ser lo más eficiente posible teniendo en cuenta las estructuras que se van a utilizar y las restricciones. El lenguaje a utilizar es el propio del diseño lógico.

Para llevarse a cabo de forma correcta el diseño lógico se han tenido en cuenta las siguientes reglas:

1. Todo tipo de entidad se convierte en una relación.
2. Todo tipo de interrelación del tipo M:N se transforma en una relación.
3. Todo tipo de interrelación del tipo 1:N se traduce en el fenómeno de propagación de claves o bien se crea una nueva relación.

Estudio de dependencias funcionales

Como parte del proceso de diseño lógico, se procedió a un estudio de dependencias funcionales con las distintas entidades que posee la base de datos.

- **numPart** → tipo ǁ IDE ǁ direccion ǁ codPostal
- **codPostal** → lugar
- **numPart** → empresa ǁ formaJuridica ǁ sede ǁ numRegPropiedad
- **numPart** → oficinaCatastro
- **numPart** → codActivacion ǁ nombre ǁ apellidos ǁ notarioEnCanton ǁ registroNotariosCanton
- **numTransac** → tipoNot ǁ expedidor_orden ǁ destinatario_orden ǁ fechaHora ǁ numRegPropiedad
- **EREID** → importe ǁ tipo ǁ fechado ǁ numDoc ǁ caseHypothecaire ǁ tasaIntMax ǁ paridadRango ǁ derechoAscRangoAnt
- **numInmueble** → EGRID ǁ municipioBarrio ǁ denominacion ǁ numCatastro ǁ hojaReg ǁ EREID_hipoteca
- **IDE_NIP** → tipo
- **IDE_NIP** → nombre ǁ fchConstitucion
- **IDE_NIP** → empresa ǁ formaJuridica ǁ sede ǁ direccion ǁ codPostal
- **IDE_NIP** → nombre ǁ apellidos ǁ fchNacimiento ǁ lugarOrigen ǁ nacionalidad ǁ sexo ǁ estadoCivil ǁ direccion ǁ codPostal ǁ NIP_comunidad
- **codPostal** → lugar
- **refInterna** → numUsuario ǁ EREID_hipoteca
- **numUsuario, refInterna** → funcion
- **numUsuario** → nombre ǁ apellidos ǁ email ǁ username ǁ password
- **numUsuario** → → telefono

- **id** → rol
- **id, numUsuario** →
- **ID** → condiciones || validoHasta
- **numPlazo** → moneda || importe || fechaValida || beneficiario || IBAN || IdPromesaPago
- **numDoc** → nombrePDF || fchConstitucion || link || numTransac
- **EREID, numPart** → fchIni || fchFin || fchFirma || IdPromesaPago
- **IDE_NIP, EREID** → rol || fchIni || fchFin || fchFirma
- **EREID, numTransac** →

Normalización

Tras haber realizado el estudio de las dependencias funcionales, se sacaron las siguientes relaciones, se procede a aplicar el proceso de normalización para normalizar las tablas candidatas de la base de datos.

- **Entidad** (numPart, tipo, IDE, direccion, codPostal, lugar)
- **Entidad_de_Credito** (numPart, empresa, formaJuridica, sede, numRegPropiedad)
- **Registro_Propiedad** (numPart, oficinaCatastro)
- **Notario** (numPart, codActivacion, nombre, apellidos, notarioEnCanton, registroNotariosCanton)
- **Transaccion** (numTransac, tipoNot, expedidor_orden, destinatario_orden, fechaHora, numRegPropiedad)
- **Hipoteca** (EREID, importe, tipo, fechado, numDoc, caseHypothecaire, tasaIntMax, paridadRango, derechoAscRangoAnt)
- **Inmueble** (numInmueble, EGRID, municipioBarrio, denominacion, numCatastro, hojaReg, EREID_hipoteca)
- **Sujeto** (IDE_NIP, tipo)
- **Comunidad** (IDE_NIP, nombre, fchConstitucion)
- **Persona_Juridica** (IDE_NIP, empresa, formaJuridica, sede, direccion, codPostal)
- **Persona_Fisica** (IDE_NIP, nombre, apellidos, fchNacimiento, lugarOrigen, nacionalidad, sexo, estadoCivil, direccion, codPostal, lugar, NIP_comunidad)
- **Referencia** (refInterna, numUsuario, EREID_hipoteca)
- **Firmante** (numUsuario, refInterna, funcion)
- **Usuario** (numUsuario, nombre, apellidos, email, telefono, username, password)
- **Rol** (id, rol)
- **UsuarioRol** (id, numUsuario)
- **Promesa_Pago** (ID, condiciones, validoHasta)
- **Plazo** (numPlazo, moneda, importe, fechaValida, beneficiario, IBAN, IdPromesaPago)

- **Documento** (numDoc, nombrePDF, fchConstitucion, link, numTransac)
- **AsignarAcreedor** (EREID, numPart, fchIni, fchFin, fchFirma, IdPromesaPago)
- **SujetoHipoteca** (IDE_NIP, EREID, rol, fchIni, fchFin, fchFirma)
- **HipotecaTransaccion** (EREID, numTransac)

1ª Forma Normal

Una tabla está en Primera Forma Normal (1FN) sí, y sólo sí, todas las columnas contiene valores atómicos, eso si, cada columna puede tener sólo un valor para cada fila de la tabla. [42] Es decir, cuando se cumpla que sus dominios no tengan elementos que a su vez sean conjuntos. [40] Basándose en esta definición se han estudiado cuales de las siguientes entidades están en 1FN y cuales han tenido que obtener un nuevo esquema de relación porque posee, al menos, un atributo multivaluado.

Las entidades que están en 1FN y por tanto normalizadas son:

- **Entidad** (numPart, tipo, IDE, direccion, codPostal, lugar)
- **Entidad_de_Credito** (numPart, empresa, formaJuridica, sede, numRegPropiedad)
- **Registro_Propiedad** (numPart, oficinaCatastro)
- **Notario** (numPart, codActivacion, nombre, apellidos, notarioEnCanton, registroNotariosCanton)
- **Transaccion** (numTransac, tipoNot, expedidor_orden, destinatario_orden, fechaHora, numRegPropiedad)
- **Hipoteca** (EREID, importe, tipo, fechado, numDoc, caseHypothecaire, tasaIntMax, paridadRango, derechoAscRangoAnt)
- **Inmueble** (numInmueble, EGRID, municipioBarrio, denominacion, numCatastro, hojaReg, EREID_hipoteca)
- **Sujeto** (IDE_NIP, tipo)
- **Comunidad** (IDE_NIP, nombre, fchConstitucion)
- **Persona_Juridica** (IDE_NIP, empresa, formaJuridica, sede, direccion, codPostal)
- **Persona_Fisica** (IDE_NIP, nombre, apellidos, fchNacimiento, lugarOrigen, nacionalidad, sexo, estadoCivil, direccion, codPostal, lugar, NIP_comunidad)
- **Referencia** (refInterna, numUsuario, EREID_hipoteca)
- **Firmante** (numUsuario, refInterna, funcion)
- **Rol** (id, rol)
- **UsuarioRol** (id, numUsuario)
- **Promesa_Pago** (ID, condiciones, validoHasta)
- **Plazo** (numPlazo, moneda, importe, fechaValida, beneficiario, IBAN, IdPromesaPago)
- **Documento** (numDoc, nombrePDF, fchConstitucion, link, numTransac)
- **AsignarAcreedor** (EREID, numPart, fchIni, fchFin, fchFirma, IdPromesaPago)

- **SujetoHipoteca** (IDE_NIP, EREID, rol, fchIni, fchFin, fchFirma)
- **HipotecaTransaccion** (EREID, numTransac)

Las entidades que no están en 1FN y que han de ser normalizadas son:

- **Usuario** (numUsuario, nombre, apellidos, email, username, password)
- **Usuario_Telefono** (numUsuario, telefono)

2ª Forma Normal

Una tabla está en la Segunda Forma Normal (2FN) sí, y sólo sí, está en 1FN y cada atributo sin clave es totalmente dependiente de la clave principal. Un atributo es totalmente dependiente de la clave principal si está en el lado derecho de una dependencia funcional, todo lo que está en el lado izquierdo es o bien la clave principal o algo sí mismo que se puede derivar de la clave principal con la transitividad de la dependencia funcional.

Se ha de destacar la definición de dependencia funcional. Es aquella propiedad de uno o más atributos único que determinan el valor de uno o más de otros atributos [42].

Las nuevas relaciones que se han obtenido en 2FN son las siguientes:

- **Entidad** (numPart, tipo, IDE, direccion, codPostal)
- **Persona_Fisica** (IDE_NIP, nombre, apellidos, fchNacimiento, lugarOrigen, nacionalidad, sexo, estadoCivil, direccion, codPostal, NIP_comunidad)
- **Lugar** (codPostal, lugar)

Las entidades que ya están en 2FN, y por tanto normalizadas.

3ª Forma Normal

Una tabla está en Tercera Forma Normal (3FN) sí, y sólo sí, para cada dependencia funcional no trivial X ->A, donde X y A son o atributos simples o compuestos, uno de los dos requisitos deben contener: cualquiera de los atributos X es una superclave o un atributo A es un miembro de una clave candidata. Si el atributo A es miembro de una clave candidata, A es llamado atributo primario. Una dependencia funcional trivial es de la forma YZ ->Z [42].

Cuando se quiere normalizar una relación a 3FN, dicha relación debe de estar en 2FN y, además, no debe de haber ninguna dependencia funcional transitiva entre los atributos que no son claves como se ha dicho anteriormente. Además, hay que tener en cuenta que las dependencias funcionales se deben de hacer para relaciones con tres o más atributos. Todas las relaciones se encuentran en 3FN [40], no se puede establecer ninguna dependencia funcional transitiva, porque todos los atributos son independientes entre sí y solamente responden ante la clave.

Forma Normal de Boyce-Codd

Una tabla R está en Forma Normal de Boyce-Codd (FNBC) si para cada dependencia funcional no trivial X ->A, X es una superclave [42].

Para que una relación se encuentre en FNBC debe de estar en 3FN y, que además no aporte información sobre atributos que no pertenezcan a una clave candidata [40]. Todas las relaciones ya se encuentran en FNBC.

4ª Forma Normal

Una tabla se encuentra en Cuarta Forma Normal (4FN) sí, y sólo sí, siempre que exista un dependencia multivaluada en R de la forma que R.x→ →R.y, todos los demás atributos de R son funcionalmente dependientes de R.x. [40].

- **Usuario_Telefono** (numUsuario, telefono)

Como se puede comprobar esta relación cumple la 4FN.

5ª Forma Normal

Una tabla se encuentra en Quinta Forma Normal (5FN) si se encuentra en 4FN sí, y sólo sí, cada dependencia está implicada por las claves candidatas y no por otro atributo [40].

- **Usuario_Telefono** (numUsuario, telefono)

Como se puede comprobar esta relación cumple la 5FN y, por tanto, normalizada.

5. Implementación de la capa de persistencia, de dominio y API

Una vez finalizadas las fases de análisis y diseño, se va a pasar a la fase de implementación y codificación. En esta fase se va a dar forma a todo lo que se ha visto en los capítulos anteriores. Es una fase un poco más extensa y compleja, ya que, se pueden encontrar muchos problemas que se deben solucionar.

En este capítulo se va a explicar la capa de persistencia, de dominio, los servicios web y la seguridad que se ha puesto en el sistema.

5.1. Implementación de la capa de persistencia y de dominio

En este apartado se va a explicar detalles de la implementación que han exigido un especial esfuerzo para construir la capa de persistencia y de dominio desde cero. Se decidió usar Hibernate, Java Persistence API (JPA) y Spring para la arquitectura básica de persistencia.

5.1.1. Hibernate

Se introdujo Hibernate como «framework» de persistencia, se utilizó Hibernate ya que es el «framework» de persistencia más conocido. Hibernate [39] es un «framework» Object Relational Mapping (ORM) cuya tarea es la de permitir a un desarrollador mapear objetos contra registros de una base de datos. Especifica reglas de mapeo utilizando XML o anotaciones. Aunque XML se utilizó inicialmente para definir reglas de mapeo, se recomienda utilizar las anotaciones ya que este enfoque es más simple y conciso. Hibernate soporta un conjunto de anotaciones orientadas a facilitar la persistencia de los objetos. Cada clase java (entidad) representa una tabla de la base de datos, se mapean los objetos poniendo dichas anotaciones que proporciona Hibernate en la clase java.

Hibernate provee de un conjunto nuevo de anotaciones orientadas a crear relaciones entre las distintas clases java y asegurar su persistencia. Se va a enumerar a continuación algunas de las anotaciones más importantes:

- @OneToMany: Anotación que se encarga de crear una relación de 1 a n entre dos clases predeterminadas.
- @ManyToOne: Anotación que se encarga de crear una relación de muchos a uno entre dos clases predeterminadas.
- @JoinColumn: Anotación que sirve para discernir qué columna de la tabla de la base de datos se usa como clave externa o foránea a la hora de aplicar la relación.
- @OneToOne: Anotación que se encarga de crear una relación de 1 a 1 entre dos clases predeterminadas.

5.1.2. Java Persistence API

JPA, es la API de persistencia desarrollada para la plataforma Java EE. Esta tarea se centró en hacer evolucionar la capa de persistencia creada con Hibernate a una nueva capa de persistencia que delega en

Hibernate pero se apoya en el API de JPA para trabajar. Por supuesto, la clase de implementación DAO utilizará Hibernate apoyándose en el API de JPA. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos (siguiendo el patrón de mapeo objeto-relacional).

5.1.3. El principio DRY y el patrón GenericDAO

Para que los objetos DAO cumplan el principio DRY (véase 4.1.1) se diseñó una nueva interfaz con las funciones que son compartidas por todas las clases de forma idéntica (borrar, salvar, buscarPorClave, buscarTodos). Se construyó una interfaz distinta a las anteriores, una interfaz tipo Genérico de tal forma que cada una de las clases definen a qué tipo hace referencia.

```
1  package com.grode.aplicacion.dao;
2
3  import java.io.Serializable;
4  import java.util.List;
5
6  public interface GenericDAO<T, ID extends Serializable> {
7          T findById(ID id);
8          List<T>findAll();
9          long countAll();
10         List<T> findEntries(int firstResult, int maxResults);
11         void update(T objeto);
12         void delete(T objeto);
13         void create(T objeto);
14 }
```

Tras crear esta interfaz, se construyó una clase genérica que implementará la interfaz, implementando los métodos de forma genérica. Al hacer uso de genéricos se pudieron eliminar una gran parte de la repetición de código que se tenía en las clases DAO de JPA. El código de esta clase genera las operaciones de persistencia elementales para todas las clases DAO. Por lo tanto, no fue necesario implementar estos métodos en las clases hijas, ya que si estas se extienden de la clase GenericDAOImpl es suficiente. Las clases JPAImpl quedaron simplificadas.

5.1.4. El principio IOC y el framework Spring

Previamente, no se necesitaba hacer uso del operador «new» sino que se apoyó en factorías. Más adelante, se querían eliminar el sistema de factorías previamente construido, añadiendo mayor flexibilidad a la aplicación en proceso de construcción, utilizando el principio de inversión de control (véase 4.1.1). El «framework» Spring es un «framework» de Inversión de Control que permite gestionar nuestras necesidades de inversión de control agrupada y sencillamente sin tener que recurrir al uso de patrones de diseño clásicos (como Factory y Abstract Factory). Los objetivos fueron la introducción al «framework» Spring y permitir cambiar las implementaciones de clases concretas de forma totalmente transparente al desarrollador, aportando flexibilidad usando el «framework» Spring.

Este «framework» trabaja como inversor de control y hace las tareas de una factoría. En este caso se trata de una factoría capaz de crear objetos apoyándose en un fichero de configuración XML (applicationContext.xml). Así, se usó un «framework» de inversión de control para poder cambiar la implementación de la capa de persistencia de una forma flexible, permitiendo tanto usar JPA, Hibernate, Java Database Connectivity (JDBC) como otras.

5.1.5. Inyección de Dependencia y Spring framework

En la parte de las implementaciones de los servicios se utilizó el principio de inyección de dependencia (véase 4.1.1), de tal forma que, a partir de este momento no sea ella la encargada de inicializar sus dependencias sino un agente externo. Se le añadió los siguientes métodos set/get a su implementación e interfaz.

```
public interface ServicioInmueble {
    InmuebleDAO getInmuebleDAO();
    void setInmuebleDAO(InmuebleDAO inmuebleDAO);
}
```

Los «frameworks» de inyección de dependencia permiten asignar las dependencias fuera de el propio código fuente, se apoya normalmente en ficheros de configuración xml. Este es el caso de Spring, que a través de su fichero xml es capaz de inyectar las distintas dependencias a los distintos objetos que se construyan.

Se ha trabajado con el «framework» Spring como inyector de dependencia a la hora de crear los servicios y las capas DAO. Tras esto, se tenía que aplicar el principio de inyección de dependencia a las propias clases DAO que se tenían construidas. Al revisar los distintos métodos de la clase GenericDAOJPAImpl, se vió que no sólo éste método si no todos los demás dependen de un objeto EntityManagerFactory. Se tuvo que refactorizar el código de la clase GenericDAOJPAImpl, así como el de su interfaz para inyectar a través de Spring dicho objeto. Se añadió set/get.

Tras ser definidos los nuevos métodos que iban a permitir inyectar la dependencia de EntityManagerFactory a las clases DAO, era momento de ver qué nuevos «bean» debían ser añadidos a nivel de fichero xml.

- DriverManagerDataSource: «Bean» encargado de crear un «pool» de conexiones contra la base de datos que se seleccione pasando usuario, «password», «url», «driver».
- EntityManagerFactory: Hace uso del DataSource creado y asigna el tipo de persistencia y algunos parámetros adicionales, como es el dialecto de JPA que se utilizará.

Las capas DAO se apoyan en Spring para inyectar la dependencia que necesitan de EntityManagerFactory. Esta dependencia necesitaba a su vez la inyección de varias propiedades y de una fuente de datos («pool» de conexiones a base de datos). Es lo que se configuró en el fichero:

```
        <bean id="dataSource"
                class="org.springframework.jdbc.datasource.
                   DriverManagerDataSource">
                <property name="driverClassName" value="com.mysql.jdbc.Driver"
                    />
                <property name="url" value="jdbc:mysql://localhost/
                   TransaccionesTerravis" />
                <property name="username" value="root" />
                <property name="password" value="nena" />
        </bean>

        <bean id="entityManagerFactory"
                class="org.springframework.orm.jpa.
                   LocalContainerEntityManagerFactoryBean">
                <property name="persistenceUnitName" value="
                   transaccionesTerravis" />
                <property name="dataSource" ref="dataSource" />
                <property name="jpaVendorAdapter">
```

```
14                          <bean class="org.springframework.orm.jpa.vendor.
                               HibernateJpaVendorAdapter">
15                               <property name="databasePlatform" value="org.
                                    hibernate.dialect.MySQL5Dialect" />
16                               <property name="showSql" value="true" />
17                          </bean>
18                  </property>
19          </bean>
```

5.1.6. Programación orientada a aspectos

Existían aún algunas mejoras que se podían realizar aunque no eran evidentes. Cada uno de los métodos de modificación ejecutaban esa operación dentro de una transacción, es decir, cada método ejecutaba una transacción. Lo que sucedía es que la responsabilidad sobre la gestión de transacciones estaba distribuida y cada uno de los métodos era responsable de gestionar sus propias transacciones. Los métodos de las clases se encargaban de gestionar de forma independiente las transacciones.

El diseño hasta ahora tenía un problema importante ya que la necesidad de que varios métodos se ejecuten de forma conjunta (Transaccional) es algo muy necesario para las aplicaciones. Para solventar este problema, se necesitó centralizar la gestión de las transacciones y no tenerla dispersa por todo el código. Para ello se debía introducir el concepto de programación orientada a aspecto o Aspect Oriented Programming (AOP).

Introducción a AOP

Los métodos de las clases se encargaban de gestionar de forma independiente las transacciones. Si se quería unificar la gestión de transacciones, se debía extraer la responsabilidad de ejecutar las transacciones de las clases. No era sencillo, si se extraía la responsabilidad de las clases, no se podría ejecutar los métodos de forma transaccional ya que no dispondrían de esa capacidad. Se solventó esto introduciendo un patrón de diseño fundamental para este tipo de programación:

Patrón «Proxy»: se utiliza como intermediario para acceder a un objeto, permitiendo controlar el acceso a él. Para ello obliga que las llamadas a un objeto ocurran indirectamente a través de un objeto «proxy», que actúa como un sustituto del objeto original, delegando luego las llamadas a los métodos de los objetos respectivos.

Usando Proxies con Spring

La capa de servicios y DAO fue creada a través del «Framework» Spring el cuál tiene la función de factoría. Al serlo, podría ser el encargado de construir un «proxy» para cada uno de los objetos creados de tal forma que sean estos «proxies» los encargados de aglutinar la responsabilidad relativa a las transacciones (funcionalidad adicional). Spring posee la capacidad de generar dinámicamente «proxies» basándose en las clases ya existentes sin tener que escribir una línea de código. Se pudo crear el conjunto de objetos que se necesitó y se les asignó a todos un «proxy» de forma automática. Una vez hecho esto, cada vez que una aplicación cliente desee acceder a un objeto determinado, el «proxy» actuará de intermediario y añadirá la funcionalidad adicional que se desee.

Ahora, se debía decidir que tipo de «proxy» se quería crear pues Spring soporta varios. En este caso, se definieron «proxies» que ayuden a gestionar las transacciones entre las distintas invocaciones a los métodos. Se eligió los «proxies» de transacción, que serán los encargados de ejecutar la funcionalidad relativa a las transacciones.

Configuración de proxies y transacciones

Para poder trabajar con «proxies» que gestionen transacciones en la aplicación se debía añadir un gestor transaccional a la configuración del «framework» Spring. En este caso, se utilizó un gestor transaccional orientado a JPA, que se define a través de un «bean» de Spring.

```
<bean id="transactionManager" class="org.springframework.orm.jpa.
   JpaTransactionManager">
        <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>
```

En segundo lugar, al fichero de configuración se debía añadir una nueva etiqueta que sería la encargada de crear los distintos «proxies» transaccionales para todas las clases.

```
<tx:annotation-driven>
```

Esta etiqueta, a nivel de fichero xml, permitirá al «framework» Spring dar de alta «proxies» para todas aquellas clases que dispongan de métodos marcados como transaccionales. Para marcar los métodos como transaccionales se tuvo que añadir un nuevo conjunto de anotaciones a esta aplicación.

@Transaccional

Así, se estaba señalando que el método se ejecuta dentro de una transacción. Estas anotaciones afectan a las clases de la capa DAO y de la capa de servicios.

```
@Transactional
        public void delete(Lugar lugar){
                lugarDAO.delete(lugar);
        }
```

5.1.7. Uso de anotaciones y principio de convención sobre configuración

Los objetivos van a ser simplificar el fichero de configuración de Spring apoyándose en anotaciones y en el principio COC (véase 4.1.1).

@PersistenceContext y Entity manager

En ese instante, se tenía diseñada nuestra capa DAO a través de JPA y utilizando la clase JPADAOSupport como apoyo a la hora de crear cualquiera de las clases DAO. Al depender todas las clases de la clase JPADAOSupport, todas soportarán la asignación de un entityManagerFactory a través del fichero applicationContext.xml de Spring.

A partir de aquí, la intención fue reducir este fichero de configuración y eliminar etiquetas. Se utilizó la anotación soportada por el estándar de JPA: la anotación @PersistenceContext. Esta anotación inyectará de forma directa un entityManager a cada una de las clases DAO de forma transparente, sin tener que usar etiquetas `<property>` a nivel del fichero applicationContext.xml. Para ello, se añadió esta nueva línea:

```
<context:annotation-config />
```

Esta línea se encarga de inyectar de forma automática el «entity manager» a todas las clases que usen la anotación @PersitenceContext. Por lo tanto, a partir de ese momento, ninguna de las clases tendrá asignada propiedades de «entity manager». Seguidamente, se muestra la modificación que se tuvo que realizar a nivel de la clase GenericDAOJPAImpl.

```
1  public abstract class GenericDAOJPAImpl<T, ID extends Serializable> implements
       GenericDAO<T, ID> {
2          private Class<T> claseDePersistencia;
3
4          @PersistenceContext
5          private EntityManager manager;
6
7          public EntityManager getManager() {
8                  return manager;
9          }
10
11         public void setManager(EntityManager manager) {
12                 this.manager = manager;
13         }
14
15         @SuppressWarnings("unchecked")
16         public GenericDAOJPAImpl(){
17                 this.claseDePersistencia = (Class<T>) ( (ParameterizedType)
                       getClass().getGenericSuperclass()).getActualTypeArguments
                       ()[0];
18         }
19
20         @Override
21         public T findById(ID id){
22                 if (id == null) return null;
23                 return (T) manager.find(claseDePersistencia, id);
24         }
25
26         @Override
27         public List<T> findAll(){
28                 try{
29                         return (manager.createQuery("select o from " +
                                   claseDePersistencia.getSimpleName()+ " o",
                                   claseDePersistencia)).getResultList();
30                 } finally{
31                         manager.close();
32                 }
33         }
34
35         @Override
36         public long countAll(){
37                 try{
38                         return (manager.createQuery("SELECT COUNT(o) FROM " +
                                   claseDePersistencia.getSimpleName() + " o", Long.
                                   class).getSingleResult());
39                 }finally{
40                         manager.close();
41                 }
42         }
43
44         @Override
45         public List<T> findEntries(int firstResult, int maxResults) {
46                 try{
47                         return (manager.createQuery("SELECT o FROM " +
                                   claseDePersistencia.getSimpleName() + " o",
```

```
                                claseDePersistencia).setFirstResult(firstResult).
                                setMaxResults(maxResults).getResultList());
48                      }finally{
49                              manager.close();
50                      }
51              }
52
53              @Transactional
54              @Override
55              public void delete(T objeto){
56                      manager.remove(manager.merge(objeto));
57              }
58
59              @Transactional
60              @Override
61              public void update(T objeto){
62                      manager.merge(objeto);
63              }
64
65              @Transactional
66              @Override
67              public void create(T objeto){
68                      manager.persist(objeto);
69              }
70      }
```

Como se puede ver se ha usado directamente la clase «manager» para invocar los métodos de persistencia y no se ha usado para nada el método getTemplate() de la clase JPADAOSupport. Esta clase se encargaba de obtener el «manager» de Spring y de gestionar las excepciones. Ya no se hereda de ella puesto que no se la necesita para inyectar el objeto EntityManager. Sin embargo, si se tenía que gestionar excepciones, una vez que se dejó de heredar JPADAOSupport, se debió añadir algo más para que todo funcione como antes.

@Repository y manejo de excepciones

Para que esta aplicación se comporte de la misma forma sin importar cuál es el «framework» de persistencia usado o cuáles son las excepciones que se lanzan, se tuvo que marcar las clases de capa DAO con la anotación @Repository, la cuál se encargará de traducir las excepciones que se hayan construido, así como de cerrar recursos. Se va a ver cómo las clases hacen uso de la anotación. A continuación se muestra el código de una de ellas. Se le incorporó a todas las clases DAO esta anotación.

```
1   @Repository
2   public abstract class GenericDAOJPAImpl<T, ID extends Serializable> implements
        GenericDAO<T, ID> {
3           private Class<T> claseDePersistencia;
```

Al marcar las clases con la anotación @Repository, se consigue que Spring gestione el manejo de excepciones de una forma más integrada, encargándose de cerrar los recursos de conexiones abiertas, etc. Ahora bien, al marcar la clase con esta anotación, ya no será necesario registrar esta clase a nivel de applicationContext.xml pues al anotarla queda registrada de forma automática. Por eso, se eliminaron las clases DAO del fichero xml.

@Service y capas de Servicio

De la misma manera se le añadió una nueva anotación a la clase de servicios denominada @Service que permitió eliminar también estas clases del fichero de configuración xml donde las tenía ubicadas. A continuación, se muestra el código fuente de una clase una vez añadida la anotación:

```
@Service(value="servicioLugar")
public class ServicioLugarImpl implements ServicioLugar{
}
```

Tras realizar esta operación, se eliminó las clases de servicio del fichero xml y se simplificó. También, se tuvo que añadir la siguiente línea de código en el fichero applicationContext.xml:

```
<context:component-scan base-package="com.grode.aplicacion.*" />
```

Esta línea se encargará de buscar en los distintos paquetes clases anotadas y registrarlas sin necesidad de que aparezcan en el fichero. Hecho esto, ya se tienen todas las clases registradas. Lamentablemente, el fichero xml no sólo se encargaba de registrarlas sino también de relacionarlas a través de la etiqueta `<property>` y esta funcionalidad todavía no se había cubierto.

Anotaciones AutoWired e inyección de dependencia

La anotación @AutoWired está basada en el principio de conversión sobre configuración para simplificar la inyección de dependencias entre las distintas clases. Se encarga de buscar las relaciones existentes entre los distintos objetos registrados e inyecta de forma automática las dependencias. Se va a ver como queda el código de la clase servicio:

```
@Service(value="servicioLugar")
public class ServicioLugarImpl implements ServicioLugar{
        @Autowired
        private LugarDAO lugarDAO;

        public LugarDAO getLugarDAO() {
                return lugarDAO;
        }

        public void setLugarDAO(LugarDAO lugarDAO) {
                this.lugarDAO = lugarDAO;
        }
}
```

De esta manera se pasó de una configuración basada exclusivamente en ficheros xml a una configuración en la que las anotaciones tienen más peso y el fichero se reduce significativamente. Las anotaciones y el principio de convención sobre configuración han permitido eliminar información redundante a nivel de la configuración. Seguidamente se muestra cómo queda el fichero applicationContext.xml, como se puede observar ha sido simplificado sin tener que registrar ninguna clase.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
```

```
http://www.springframework.org/schema/context/spring-context-3.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
"
        xmlns:tx="http://www.springframework.org/schema/tx" xmlns:context="
           http://www.springframework.org/schema/context">

        <tx:annotation-driven />
        <context:annotation-config />
        <context:component-scan base-package="com.grode.aplicacion.*" />
        <bean id="transactionManager" class="org.springframework.orm.jpa.
           JpaTransactionManager">
                <property name="entityManagerFactory" ref="
                   entityManagerFactory" />
        </bean>

        <bean id="dataSource"
                class="org.springframework.jdbc.datasource.
                   DriverManagerDataSource">
                <property name="driverClassName" value="com.mysql.jdbc.Driver"
                    />
                <property name="url" value="jdbc:mysql://localhost/
                   TransaccionesTerravis" />
                <property name="username" value="root" />
                <property name="password" value="nena" />
        </bean>

        <bean id="entityManagerFactory"
                class="org.springframework.orm.jpa.
                   LocalContainerEntityManagerFactoryBean">
                <property name="persistenceUnitName" value="
                   transaccionesTerravis" />
                <property name="dataSource" ref="dataSource" />
                <property name="jpaVendorAdapter">
                        <bean class="org.springframework.orm.jpa.vendor.
                           HibernateJpaVendorAdapter">
                                <property name="databasePlatform" value="org.
                                   hibernate.dialect.MySQL5Dialect" />
                                <property name="showSql" value="true" />
                        </bean>
                </property>
        </bean>

</beans>
```

5.2. Implementación de los servicios web

En este apartado se va a explicar las distintas tecnologías que se han utilizado para implementar la arquitectura SOA.

5.2.1. Introducción a los servicios web

Un servicio web es definido por el W3C como [43]:

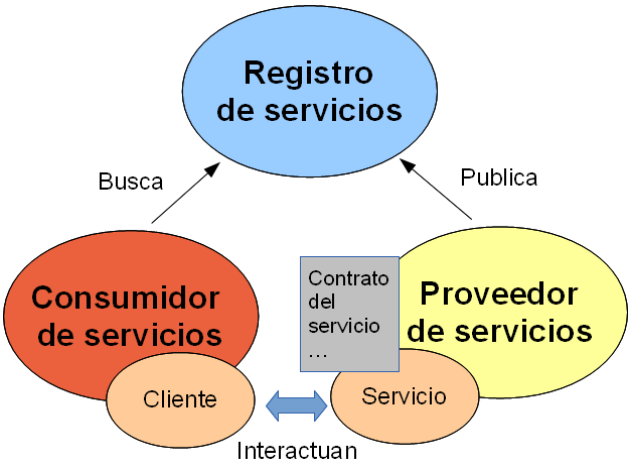


Figura 5.1.: Arquitectura de los servicios web.

> "Un servicio web es un sistema software diseñado para ofrecer interacción máquina-a-máquina sobre una red. Tiene una interfaz descrita en un formato procesable por la máquina (específicamente WSDL). Otros sistemas interactúan con el servicio web en una forma prescrita por su descripción utilizando mensajes SOAP, normalmente transmitidos usando HTTP con una serialización XML y con otros estándares web relacionados."

Simplemente, un servicio web es [44] un componente de software que proporciona una función de negocio como un servicio a través de Internet que se puede acceder a través de una URL. El mayor beneficio que proporcionan los servicios web es la interoperabilidad. Los servicios web se pueden trasladar a cualquier plataforma y se pueden escribir en diferentes lenguajes de programación. Del mismo modo, el cliente de acceso al servicio web puede ser una aplicación escrita en un lenguaje diferente y que se ejecuta en una plataforma diferente a la de un servicio en sí.

5.2.2. Desarrollo de los servicios web

Dos de los enfoques más utilizados para el desarrollo los servicios web son SOAP y REST. Este proyecto se ha desarrollado con SOAP, por lo tanto se centra en él.

Un servicio web implica tres tipos de funciones: un consumidor de servicios, un proveedor de servicios, y un registro de servicio opcional. La figura 5.1 muestra la interacción entre el proveedor de servicios, el consumidor de servicios y el registro de servicios. Los proveedores de servicios proporcionan los servicios a través de Internet y atienden las solicitudes de los servicios web. El consumidor de servicios consume los servicios ofrecidos por el proveedor de servicios. En los servicios web basados en SOAP, el proveedor de servicios publica el contrato (archivo WSDL) del servicio a través de la web en la que un consumidor puede acceder a el directamente o busca en un registro de servicios que proporciona un mecanismo para buscar servicios web. El consumidor de servicios por lo general genera un código de cliente de servicios web desde un archivo WSDL utilizando las herramientas que ofrece el marco de servicios web para interactuar con el servicio web.

Si sus requisitos consisten en varios contratos por definir y negociar entre el proveedor y el consumidor, tales como el uso de un archivo WSDL y adherirse a varias especificaciones de servicios web, tales como, la seguridad de servicios web para la adaptación empresarial, la opción correcta son los servicios web basados en SOAP. Por lo tanto, se ha usado XML Schema (Véase 1.4.5), se empleará para definir la estructura de documentos XML y SOAP (Véase 1.4.4), que permite comunicar dos tecnologías diferentes usando XML como estándar a la hora de transferir información entre los dos sistemas, utilizado conjuntamente con WSDL (Véase 1.4.3), para crear los distintos mensajes SOAP.

Una de las ventajas de usar tecnología Java a la hora de construir servicios web es que permite construir un servicio web a partir de una sencilla clase Java a través del uso de anotaciones. Estas anotaciones están definidas en el estándar Java API for XML Web Services (JAX-WS) de JEE. Una vez se anote la clase, JAX-WS se encargará de construir los «proxies» y el fichero WSDL de forma automática.

Antes, se tuvo que construir nuevas clases que realizarán la función de contenedor de información y encajarán con los campos solicitados. Estas clases se denominan habitualmente Data Transfer Objects (DTO) ya que son clases construidas especialmente para la transmisión de información entre dos sistemas. Es un objeto que por definición se envía y recibe dentro de un servicio, es decir, la información que se envía/recibe del DAO se "empaqueta" en estos objetos. Una vez construidas dichas clases, se podía proceder a realizar los servicios web.

Teniendo claro qué interfaces y clases se iba a construir, era el momento de introducir un par de anotaciones adicionales que pertenecen al estándar JAX-WS y son las encargadas de simplificar la publicación de servicios web en este tipo de entorno. JAX-WS [44] es una especificación diseñada para simplificar la construcción de servicios web basados principalmente en SOAP y clientes de servicios web en Java. JAX-WS también incluye Java Architecture for XML Binding (JAXB) y SOAP with Attachments API for Java (SAAJ). JAXB ofrece capacidades de enlace de datos, proporcionando una forma cómoda de mapear XML Schema a una representación en código Java. El JAXB protege la conversión de mensajes XML Schema en mensajes SOAP a código Java sin que los desarrolladores tengan que analizar el XML y SOAP. La especificación JAXB define la unión entre Java y el XML Schema. SAAJ proporciona una forma estándar de tratar con archivos adjuntos XML contenidos en un mensaje SOAP.

- @WebService: Anotación que marca una clase como servicio web.
- @WebMethod: Anotación que marca un método como método público a nivel del servicio web.

Se utilizan estas anotaciones para crear los servicios web necesarios. Los primeros que se realizaron fueron los servicios de entidad, un servicio web de este tipo por cada clase de entidad que haya. Es evidente que se apoyó en las anotaciones @Autowired y @Service para acceder a la información. Una vez se dispone de esta información se ha usado la anotación @WebService para publicar esta información hacia el exterior de la aplicación.

Al haber anotado la clase con @WebService JAX-WS se encargará de generar los «proxies» y el fichero WSDL necesario para acceder al servicio. Tras terminarse de construir el servicio web se debía de realizar una serie de pasos en cuanto a TomEE y Spring se refiere para poder generar los «proxies» y el fichero WSDL.

5.2.3. Apache CXF

Apache Celtix/XFire (CXF) [44] es un «framework» de servicios web de código abierto que está diseñado para ser fácil de usar, modelo de programación basado en estándares para el desarrollo de servicios web. Los servicios web se pueden implementar utilizando diferentes protocolos de aplicación como SOAP, XML, JavaScript Object Notation (JSON), REST HTTP, y soportan varios protocolos de transporte como HTTP o Java Message Service (JMS).

Se utiliza Apache CXF como «framework» de servicios web, ya que se integra muy fácilmente con Spring. Se ha elegido CXF en lugar de otros «frameworks» de servicios web, ya que es compatible con todos los principales estándares de servicios web y proporciona un modelo de programación simplificado para el desarrollo de servicios web basados en SOAP y RESTful, junto con opciones para otros protocolos de aplicación. CXF proporciona un modelo de implementación flexible para la implementación de servicios web.

Los estándares de servicios web definen las normas de una aplicación de servicio web con respecto a su interoperabilidad. Las normas garantizan que se accede a un servicio web independientemente de la plataforma del cliente. El «framework» proporciona soporte al estándar JAX-WS, es una de las tecnologías de servicios web más importantes. Las normas y especificaciones se aplican a un archivo WSDL, dicho archivo sirve como contrato entre el proveedor de servicios y un consumidor de servicios en los servicios web basados en SOAP.

5.3. Seguridad en el sistema

Nunca se debe almacenar las contraseñas en claro. No se debe utilizar una función «hash» simple como Message-Digest Algorithm 5 (MD5) o Secure Hash Algorithm (SHA), estos algoritmos de «hash» estándar son rápidos y pueden ser fácilmente utilizados para probar miles de contraseñas en paralelo en un hardware personalizado. Un algoritmo «hash» es una función unidireccional que mediante datos de entrada produce una conjunto de datos de salida de longitud fija (el «hash»), una contraseña. Las características de estos algoritmos que resulta útil para gestionar las contraseñas son:

- Son “unidireccionales”: es muy difícil (efectivamente imposible) obtener la entrada original partiendo del valor del «hash». Esta propiedad hace a los valores «hash» muy útiles para fines de autenticación.
- Son “deterministas”: la misma entrada siempre producirá la misma secuencia de salida.

Un problema con el uso de métodos «hash» es que es relativamente fácil de conseguir si se utiliza como entrada una palabra común. Las personas tienden a elegir contraseñas similares y están disponible en internet. De manera similar, un atacante puede construir un diccionario de «hashes» de una lista de textos estándar y utilizar este para buscar la contraseña original. Es decir, anteriormente sólo con encriptar las contraseñas utilizando métodos «hash» se podía ofrecer cierto nivel seguridad, pero actualmente esto ya no es aplicable debido a los equipos que han aumentado su potencial de procesamiento y a la aparición de técnicas de descifrado.

Una manera de prevenir esto es la de utilizar una sal al calcular los «hashes». Esta es una cadena adicional de datos para cada usuario que se combina con la contraseña antes de calcular el «hash». Lo ideal sería que los datos sean lo más aleatorios posibles, pero en la práctica cualquier sal es mejor que nada. El uso de una sal significa que un atacante tiene que construir un diccionario separado de «hashes» para cada sal, lo que hace el ataque más complicado (pero no imposible). Siempre utiliza un algoritmo «hash» de contraseñas de un sólo sentido, como *bcrypt* [45] que incorpora un token único denominado sal (bits aleatorios que se usan como parte de una clave en un cifrado), diferente para cada contraseña almacenada. Bcrypt está diseñado para ser lento y obstaculizar el descifrado de contraseñas. Usar org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder es una buena opción para la seguridad. Bcrypt genera automáticamente un valor aleatorio (16 bytes) de sal para cada contraseña cuando se codifica, y lo almacena en la cadena bcrypt en un formato estándar.

6. Implementación de la capa de composiciones

En este capítulo, se detalla una introducción a los nuevos conceptos aprendidos llegados a esta fase y se recogen las distintas composiciones que se han realizado. Se detalla, para cada una de ellas, la función que realiza, el comportamiento, los participantes que intervienen y los formularios que han sido realizados.

6.1. Aprendizaje de nuevos conceptos

Llegada a esta etapa del proyecto, fue necesario aprender nuevos conceptos necesarios para el desarrollo de las composiciones con Intalio.

6.1.1. XForms

XForms [46] es un lenguaje basado en XML para formularios Web. XForms no es un tipo de documento independiente, pero está diseñado para ser integrado en otros lenguajes de marcado, tales como Extensible Hypertext Markup Language (XHTML), Open Document Format (ODF) o Scalable Vector Graphics (SVG). Se apoya en otros estándares del W3C como XML Schema, XPath y XML Events. Un formulario web basado en XForms recoge y procesa los datos XML utilizando una arquitectura que separa la presentación, propósito y contenido. Los datos subyacentes de un formulario se organizan en instancias de esquema de datos. Un formulario XForm permite el procesamiento de datos que se producen utilizando tres mecanismos:

- Un modelo declarativo compuesto de fórmulas para los cálculos y las limitaciones de los datos, tipo de datos y otras declaraciones de propiedades y parámetros de presentación de datos.
- Una capa de visualización compuesta de controles de interfaz de usuario basadas en la intención.
- Un controlador imperativo para la orquestación de manipulaciones de datos, interacciones entre las capas de modelo y vista, y la presentación de datos.

Por lo tanto, XForms acomoda la reutilización de los componentes de los formularios, fomenta una fuerte validación de tipo de datos, elimina innecesarios viajes de ida y vuelta al servidor, ofrece independencia de dispositivo y reduce la necesidad de «scripting».

XForms también puede ser usado a través de varias tecnologías de servidor que convierten el código de XForms a formularios de HyperText Markup Language (HTML) en tiempo de ejecución y de manera transparente. Estas implementaciones incluyen a los proyectos de código abierto Chiba (ahora BetterForm) y Orbeon.

El editor de formularios Xforms es un componente de BPMS Workflow embebido en Intalio BPMS Designer. Permite crear simplemente formularios XForms arrastrando componentes desde una paleta. Además, genera automáticamente el código para desplegar los formularios en Intalio BPMS Workflow Web User Interface, integrando los mismos con los procesos.

6.1.2. XQuery 1.0

XQuery [17] es un lenguaje para la búsqueda y extracción de los elementos y atributos de documentos XML. XQuery 1.0 es una extensión de XPath 2.0. XQuery 1.0 y XPath 2.0 comparten el mismo modelo de datos [23] y soportan las mismas funciones y operadores [22]. XQuery es una recomendación del W3C y es compatible con varios estándares del W3C, como XML, espacio de nombres, Extensible Stylesheet Language Transformations (XSLT), XPath y esquemas XML.

XQuery utiliza expresiones XPath para acceder a determinadas partes del documento XML. Añade, además, expresiones similares a las sentencias SELECT de Structured Query Language (SQL), conocidas como expresiones For, Let, Where, Order by, Return (FLWOR).

- **La cláusula «for».** Se une a una variable cada elemento devuelto por la expresión. La cláusula «for» resulta en iteración. No puede haber múltiples cláusulas en la misma expresión FLWOR.
- **La cláusula «let».** Permite asignaciones de variables y evita repetir la misma expresión muchas veces. La cláusula «let» no resulta en iteracción.
- **La cláusula «where».** Se utiliza para especificar uno o más criterios para el resultado.
- **La cláusula «order by».** Se utiliza para especificar el orden de clasificación de los resultados.
- **La cláusula «result».** Especifica lo que ha de ser devuelto.

XQuery también admite expresiones condicionales IF-THEN-ELSE. Esta cláusula es útil para dar un formato diferente a la salida dependiendo de la información de ésta, es decir, se puede cambiar la estructura de los nodos en los que se recupera la información según convenga por su contenido. Además, añade la posibilidad de construcción de documentos XML a partir de los resultados de la consulta. Las expresiones Xquery devuelven los elementos de la misma manera como se describen en el documento de entrada pero se puede añadir nuestros propios elementos y atributos con el resultado.

El lenguaje se basa en el modelo en árbol de la información contenida en el documento XML. En XQuery hay siete tipos de nodos: elementos, atributos, nodos de texto, espacios de nombres, instrucciones de procesamiento, comentarios y nodos del documento (raíz). La raíz del árbol es el nodo del documento (o nodo raíz). El sistema de tipos usado por el lenguaje considera todos los valores como secuencias, asumiéndose un valor simple como una secuencia de un solo elemento. Los elementos de una secuencia pueden ser valores atómicos o nodos. Los tipos de datos primitivos o atómicos son los mismos que los de XML Schema.

Además de soportar las mismas funciones que XPath 2.0, añade funciones personalizables. Si no se encuentra la función XQuery que se necesita, se puede escribir una propia. Las funciones definidas por el usuario se pueden definir en la consulta o en una biblioteca separada.

XQuery ha sido de utilidad en este PFC porque XForms no permite introducir n veces algunos campos de los formularios y por lo tanto, se ha necesitado transformar un fragmento XML (trozo de un documento XML) a otro fragmento XML.

6.2. Detalles de implementación

En la implementación de las composiciones se ha de destacar un detalle importante a tener en cuenta:

- **Fallo Orchestration Director Engine (ODE):** El motor de Intalio no detecta los namespace de XQuery. La incidencia que ocurre es ODE-519, en `http://osdir.com/ml/dev.ode.apache.org/2009-02/msg00331.html`.

  Se produce al registrar dos AsignarAcreedor, el primero con fchFin y el segundo sin ella. Se ha utilizado una función ode:delete($asignarAcreedorServiceCreateMsg.parameters, $asignarAcreedorServiceCreateMsg.parameters/fchFin).

6.3. Composición «Registrar cédula hipotecaria»

6.3.1. Función que realiza

Hasta ahora, las cédulas hipotecarias se encontraban escritas en papel (cédulas nominativas o al portador). Con esta composición, se permite que las cédulas hipotecarias podrán ser registradas, en cualquier momento, directamente por la oficina del registro de la propiedad afectado sin ser previamente hecha una escritura pública notarial. La ley exige sin embargo el acuerdo escrito del propietario afectado que se va a realizar mediante formularios. Un proceso debe ser iniciado para cada cédula hipotecaria. Esto significa que habrá que establecer una declaración de acuerdo del propietario para cada cédula hipotecaria.

El proceso Registrar Cédula Hipotecaria se aplica en las siguientes formas:

- Se aplica cada vez que una hipoteca deba ser registrada.
- Registrar Cédula Hipotecaria se integra como un subproceso en el proceso principal Recompra de créditos entre entidades de crédito / Cambio de acreedor.

6.3.2. Participantes

Los participantes del proceso Registrar Cédula Hipotecaria son:

- Oficinas de registro de la propiedad.
- Los acreedores garantizados, es decir, entidades de crédito (bancos, seguros, cajas de pensiones).

6.3.3. Variantes

Los siguientes requisitos se aplicarán para el proceso Registrar Cédula Hipotecaria:
Proceso 1a Condiciones - Variante electrónica

- Registrar cédula hipotecaria.
- La entidad de crédito afectada está conectada al sistema.
- La oficina del registro de la propiedad afectado está conectado al sistema.
- Si se trata de un aval colectivo, no puede ser procesado electrónicamente sólo si los edificios se encuentran en el mismo distrito del registro de la propiedad.

6.3.4. Formularios

Para esta composición hay que utilizar los formularios: declaración de acuerdo, envío de notificación, orden entidad de crédito (Orden transformación cédula hipotecaria) y orden oficina de registro de la propiedad (Confirmación inscripción en el registro de la propiedad). El propietario inmobiliario así como la entidad de crédito firman en el mismo formulario.

6.3.5. Comportamiento de la composición

A continuación, se procede a describir la estructura de la composición. El usuario de la entidad de crédito inicia el proceso.

- Si ha introducido previo.

  Se rellenan las declaraciones del acuerdo para una persona física o jurídica hasta introducir continuar.

- Se introducen los números del inmueble o EGRID para comprobar que el inmueble no existen en BD hasta introducir continuar.

  Si existe, se notifica el motivo de la finalización del proceso. Si no, continúa con el proceso.

- Completa el formulario "orden entidad de crédito" y se la envía al sistema. Se comprueba si los inmuebles están en el mismo distrito que el registro de la propiedad hasta introducir continuar.

  Si no, el sistema envía una notificación y el motivo de la finalización del proceso.

- Se van introduciendo propietarios, ya sea comunidad, persona física o jurídica, y referencias hasta introducir continuar y se obtienen los números del usuario. Si se introdujo directo, se realiza en este momento la declaración del acuerdo. Se obtiene el número de transacción y los números de los participantes del proceso. El usuario completa el formulario "Envío de notificación" y se lo envía al sistema. El usuario del registro de la propiedad recibe los documentos y completa la orden. El sistema comprobará si se puede inscribir la hipoteca en la BD.

  Si es así, la hipoteca se registra en la base de datos. Además de registrarse asignar acreedor y todos los sujetos, los inmuebles, las referencias y los firmantes introducidos anteriormente.

- El sistema comprobará si se puede inscribir la transacción en la BD.

  Si es así, el sistema registra la transacción en la BD.

- Si alguna de las dos comprobaciones anteriores es no, envía una notificación y el motivo de la finalización del proceso. El sistema envía una notificación a los participantes del proceso para la confirmación de la inscripción. Si ocurriera cualquier excepción durante el proceso, se captura y se notifica enviando un mensaje indicando el fallo producido.

En las figuras 6.1 y 6.2, se puede ver el diagrama BPMN del proceso Registrar cédula hipotecaria completo, es una representación esquemática de lo que se ha hecho en Intalio. Se muestra en su nivel más alto de abstracción, el flujo del proceso muestra sólo las actividades más importantes. Cada una de las actividades del nivel superior de abstracción se descomponen en flujos detallados. El proceso se va descomponiendo hasta que se alcanzan actividades atómicas (es decir, actividades que no se pueden descomponer aún más). En la figura 6.3 se puede observar una parte del comportamiento de este proceso en el editor de Eclipse en el nivel más bajo de abstracción, el código BPEL generado. Esta última figura, se refiere a parte del contenido del subproceso "Registrar los datos de la orden y de la transacción" del diagrama BPMN.

6.4. Composición «Cambio de acreedor»

6.4.1. Función que realiza

La cesión de los derechos del acreedor sobre una cédula hipotecaria de registro es efectuada por uno u otro acreedor por medio de un requerimiento de cesión ante la oficina del registro de la propiedad afectado dependiendo de:

- Cédula hipotecaria Nominativa en papel. La Requisa de inscripción del nuevo acreedor en el registro de acreedores será efectuada por el nuevo acreedor (la solicitud de registro de un nuevo acreedor en el registro de acreedores se lleva a cabo por el nuevo acreedor).

- Cédula hipotecaria de registro. La Requisa de inscripción del nuevo acreedor al registro de la propiedad es efectuada por el antiguo acreedor (la solicitud de registro de un nuevo acreedor en el registro de la propiedad se lleva a cabo por el anterior acreedor).

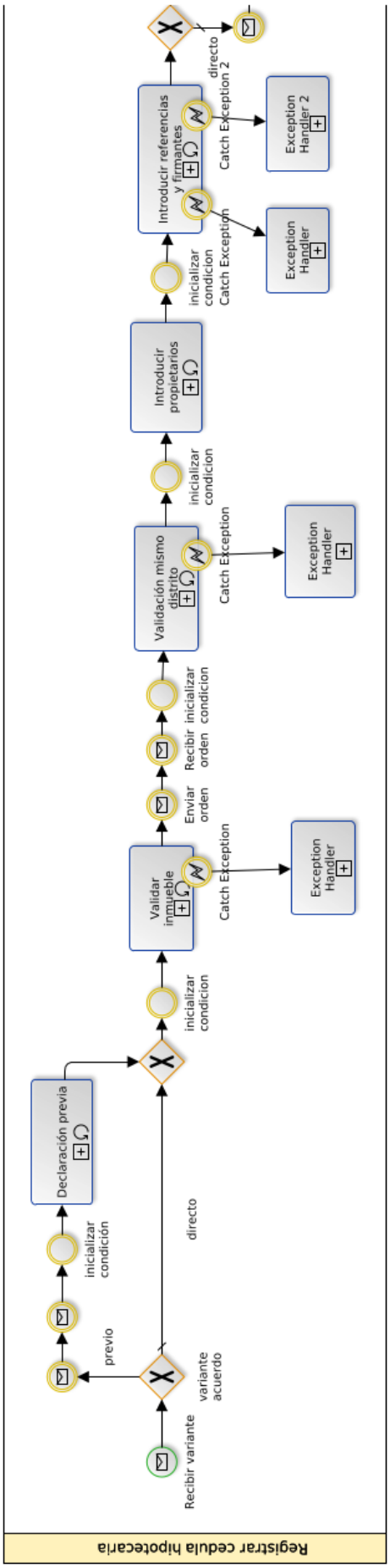


Figura 6.1.: Diagrama BPMN parte 1 - Registrar cédula hipotecaria.

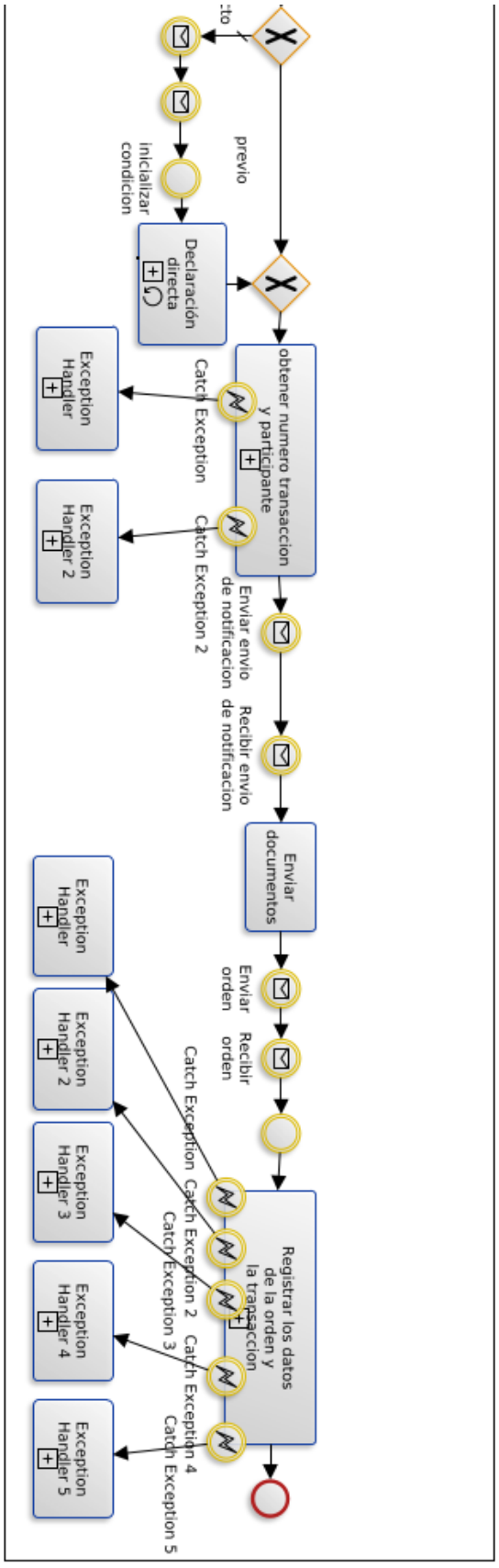


Figura 6.2.: Diagrama BPMN parte 2 - Registrar cédula hipotecaria.

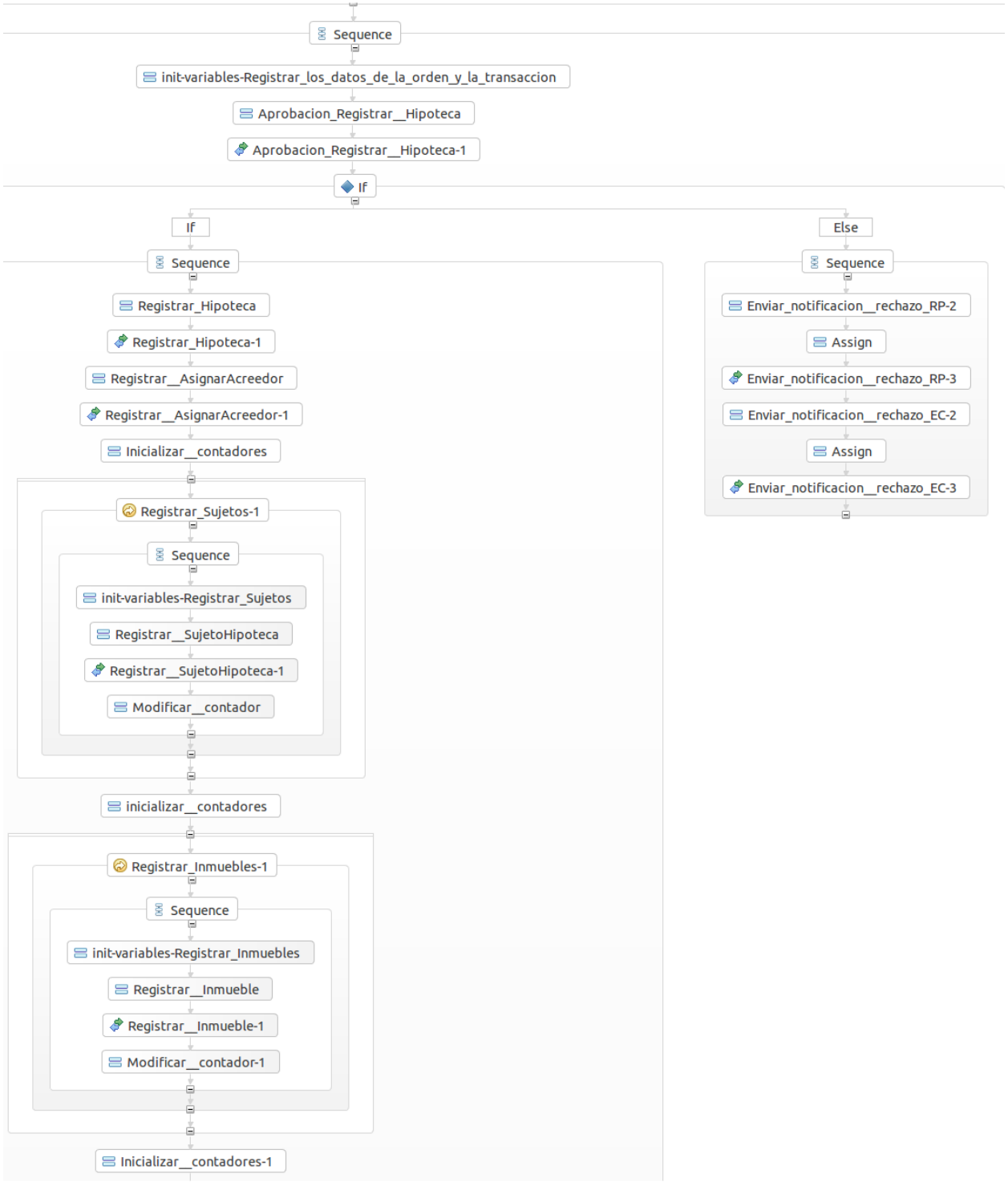


Figura 6.3.: Parte del código BPEL autogenerado por Intalio de Registrar cédula hipotecaria.

La recompra de una hipoteca es en principio una cuestión de derecho privado que se realiza entre el tomador de crédito/prestatario/deudor, la entidad de crédito vendedora (antigua) y la entidad de crédito compradora (nueva). El plazo de liquidación se elevó de 12 a 18 meses. Los requisitos necesarios para el cambio de acreedor son:

- Los procesos siguientes deberán ser utilizados cada vez que una entidad de crédito adquiriese un crédito hipotecario a otra entidad de crédito.
- Son considerados como entidades de crédito los bancos, seguros y las cajas de pensiones.
- El proceso es en principio iniciado por la entidad de crédito compradora (nueva).
- En el marco de los procesos 2A, el rescate del crédito se acompaña de la transformación de la cédula hipotecaria sobre papel en cédula hipotecaria de registro. En caso de que la entidad de crédito compradora desea renunciar a la transformación de la cédula hipotecaria sobre papel, entonces debería elegir el proceso 2.
- Una transacción puede referirse a varias cédulas hipotecarias y una cédula hipotecaria varios inmuebles, siempre que todas las cédulas hipotecarias y todos los edificios relacionados con el mismo distrito del catastro, es decir, la misma oficina del registro de la propiedad.
- En la transacción, en el marco de una promesa de pago condicional e irrevocable pueden ser tratadas varias cuotas de pago, éstas antes no obstante tienen entre ellas un enlace lógico (mismo deudor).

6.4.2. Participantes

Los participantes del proceso Recompra de créditos entre entidades de crédito / cambio de acreedor son:

- Las entidades de crédito en el marco de su actividad de prestamista y de acreedor garantizado (bancos, seguros, cajas de pensiones).
- Oficinas del registro de la propiedad.

6.4.3. Variantes

Los siguientes requisitos se aplicarán para el proceso de nueva compra de créditos entre entidades de crédito / cambio de acreedor:

Proceso 2a. Condiciones

- Registrar cédula hipotecaria.
- La oficina del registro de la propiedad afectado está conectada al sistema.
- La entidad de crédito compradora (nueva) es un participante del sistema.
- La entidad de crédito vendedora (antigua) es un participante del sistema.

Proceso 2c

- La cédula hipotecaria de registro está ya a disposición.
- La oficina del registro de la propiedad afectado está conectada al sistema.
- La entidad de crédito compradora (nueva) es un participante del sistema.
- La entidad de crédito vendedora (antigua) es un participante del sistema.

6.4.4. Formularios

Para los procesos hay que utilizar los formularios: Notificación promesa de pago condicional e irrevocable, requerimiento electrónico de cambio de acreedor y confirmación electrónica de la inscripción al registro de la propiedad. En el marco del proceso Recompra de créditos entre entidades de crédito / cambio de acreedor, la entidad de crédito compradora (nueva) le hace a la entidad de crédito vendedora (antigua) una promesa de pago condicional e irrevocable, que contiene información sobre la entidad de crédito compradora (nueva), información sobre la entidad de crédito vendedora, información sobre el objeto, información sobre el deudor, promesa de pago condicional e irrevocable, precisiones sobre el pago, enumeración de las condiciones y plazo.

6.4.5. Comportamiento de la composición

A continuación, se procede a describir la estructura de la composición. El usuario de la entidad de crédito compradora (nueva) inicia el proceso y el sistema recibe mediante un mensaje la variante inicial.

- Si es sobre papel (a).
  - Si el usuario ha introducido previo para la declaración del acuerdo.

    Se rellenan las declaraciones del acuerdo para una persona física o jurídica hasta introducir continuar.
  - Se introducen los números del inmueble o EGRID para comprobar que el inmueble no existen en BD hasta introducir continuar.

    Si existe, se notifica el motivo de la finalización del proceso. Si no, continúa con el proceso.
  - El usuario añade los datos que faltan al formulario “orden Entidad de Crédito” y se la envía al sistema. Comprueba si los inmuebles están en el mismo distrito que el registro de la propiedad hasta introducir continuar y se comprueba que las dos entidades de crédito tengan el mismo registro de la propiedad.

    Si no, envía una notificación y el motivo de la finalización del proceso.
  - Se van introduciendo propietarios, ya sea comunidad, persona física o jurídica, y referencias hasta introducir continuar y se obtienen los números del usuario.
- Si es de registro (c).
  - El usuario de la entidad de crédito vendedora rellena un formulario con los datos del cambio de acreedor. El sistema comprueba que el IDE de la entidad de crédito que financiaba hasta ahora es el de la entidad de crédito vendedora y que está registrada, y además, se comprueba que las dos entidades de crédito tengan el mismo registro de la propiedad.

    Si no, envía una notificación y el motivo de la finalización del proceso.
- Si es cualquiera de los dos (a, c).
  - El sistema recibe mediante un mensaje una promesa de pago de la entidad de crédito compradora a favor de la vendedora (antigua) y se van introduciendo todos los plazos que se desee hasta introducir continuar.

    Si es aceptada, el sistema envía la confirmación realizada por la entidad de crédito vendedora a la compradora mediante un mensaje y se valida la fecha de los plazos que debe ser de menos de 18 meses. Si es rechazada, se vuelve a repetir el proceso hasta ser aceptada.
- Si es sobre papel (a).

- Se obtiene el número de la transacción y los números de los participantes del proceso. Si al inicio se introdujo directo, se realiza en este momento la declaración del acuerdo. El usuario de la entidad de crédito, el cuál, completa el formulario "Envío de notificación" y se lo envía al sistema. El usuario del registro de la propiedad recibe los documentos y completa la orden. El sistema comprobará si la inscripción de la hipoteca en la BD puede ser efectuada.

  Si es así, la hipoteca y el cambio de acreedor se registran en la base de datos. Se registran en la BD todos los sujetos, los inmuebles, las referencias y los firmantes introducidos anteriormente. Se registra la promesa de pago, los plazos y asignar acreedor para dichas entidades de crédito.

- Si es de registro (c).
  - El sistema comprobará si se puede inscribir el cambio de acreedor en la BD.

    Si es así, la hipoteca se actualiza en la base de datos, se registra la promesa de pago, los plazos y asignar acreedor para dichas entidades de crédito.

- Si es cualquiera de los dos (a, c).
  - El sistema comprobará si se puede inscribir la transacción en la BD.

    Si es así, el sistema registra la transacción en la BD.
  - Si alguna de las dos comprobaciones anteriores es no, envía una notificación y el motivo de la finalización del proceso. El sistema envía una notificación a los participantes del proceso para la confirmación de la inscripción. Si ocurriera cualquier excepción durante el proceso, se captura y se notifica enviando un mensaje indicando el fallo producido.

En las figuras 6.4 y 6.5, se puede ver el diagrama BPMN del proceso Cambio de acreedor, es una representación esquemática de lo que se ha hecho en Intalio. Se muestra en su nivel más alto nivel de abstracción, el flujo del proceso muestra sólo las actividades más importantes. Cada una de las actividades del nivel superior de abstracción se descomponen en flujos detallados. El proceso se va descomponiendo hasta que se alcanzan actividades atómicas (es decir, actividades que no se pueden descomponer aún más). En la figura 6.6 se puede observar una parte del comportamiento de este proceso en el editor de Eclipse en el nivel más bajo de abstracción, el código BPEL generado. Esta última figura, se refiere a parte del contenido del subproceso "Registrar los datos del cambio del acreedor y la transaccion" del diagrama BPMN.

6.5. Composición «Constitución de cédulas hipotecarias de registro»

6.5.1. Función que realiza

El registro de la cédula hipotecaria se inicia con su inscripción en el registro de la propiedad y no podrá estar constituida ya que no ha sido creado ningún título hipotecario por las cédulas hipotecarias de registro. La Constitución de la cédula se hará bien mediante un contrato de garantía de dos páginas o de una declaración de una página. Es posible constituir una cédula hipotecaria de registro por contrato de garantía.

6.5.2. Participantes

Los participantes del proceso Constitución de cédulas hipotecarias de registro son:

- Las entidades de crédito en el marco de su actividad de prestamista y de acreedores garantizados (bancos, seguros, cajas de pensiones).
- Los funcionarios públicos.
- Las oficinas del registro de la propiedad.

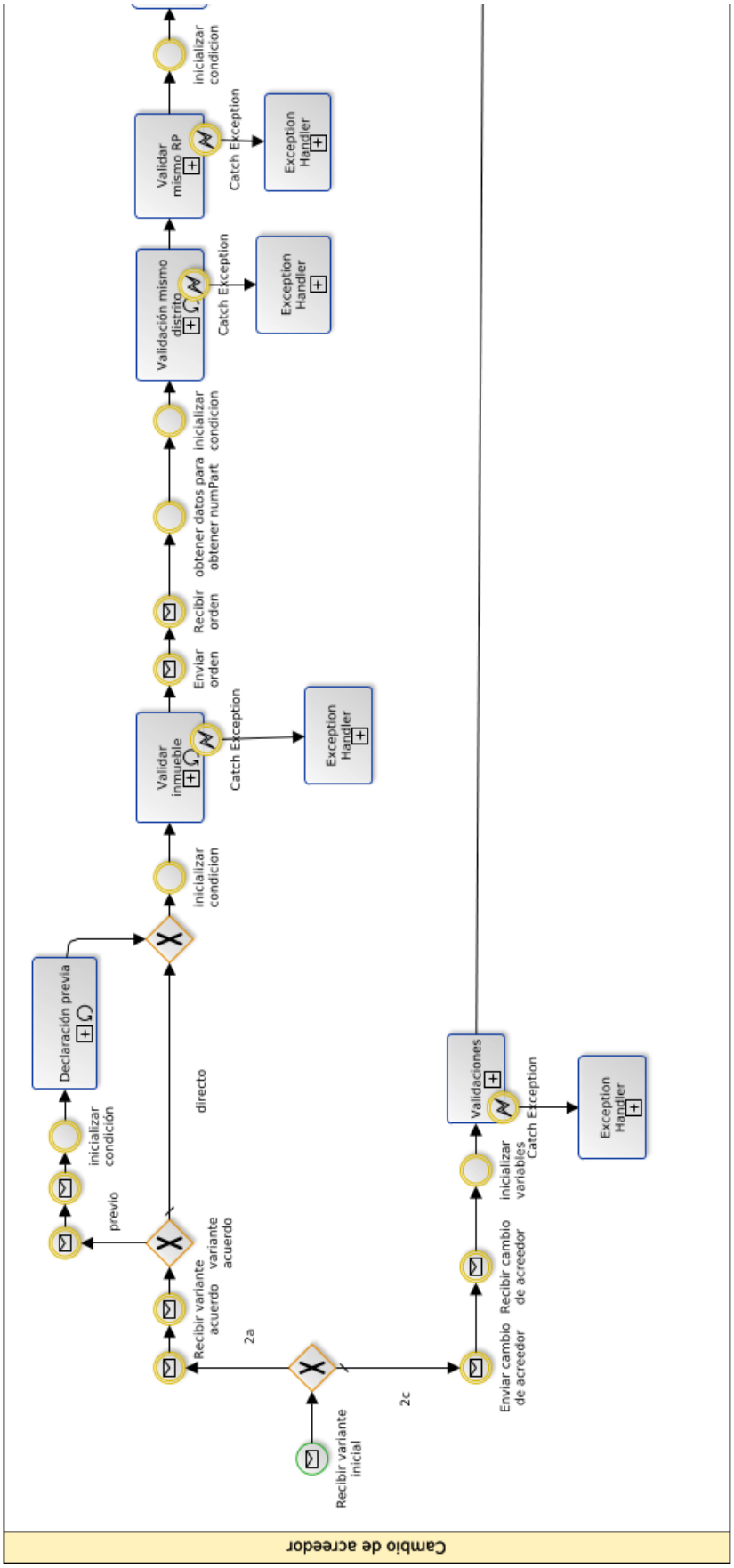


Figura 6.4.: Diagrama BPMN parte 1- Cambio de acreedor.

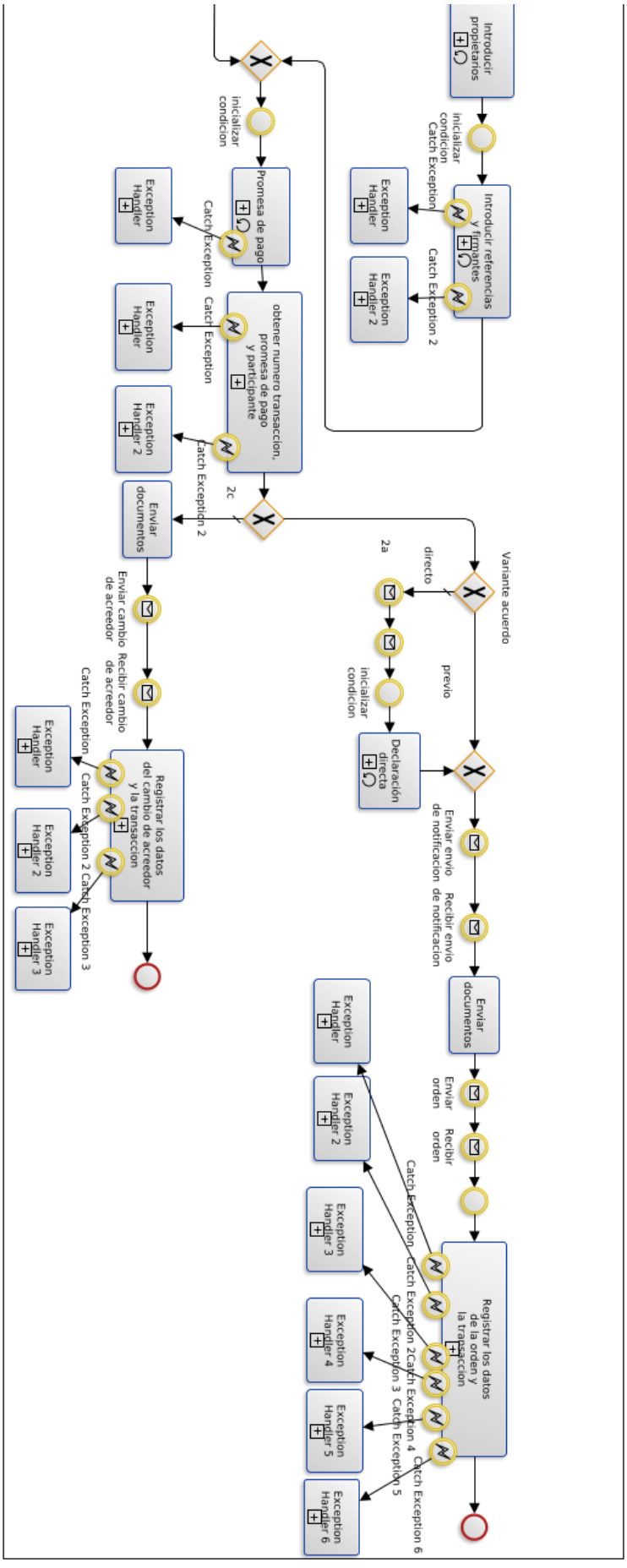


Figura 6.5.: Diagrama BPMN parte 2- Cambio de acreedor.

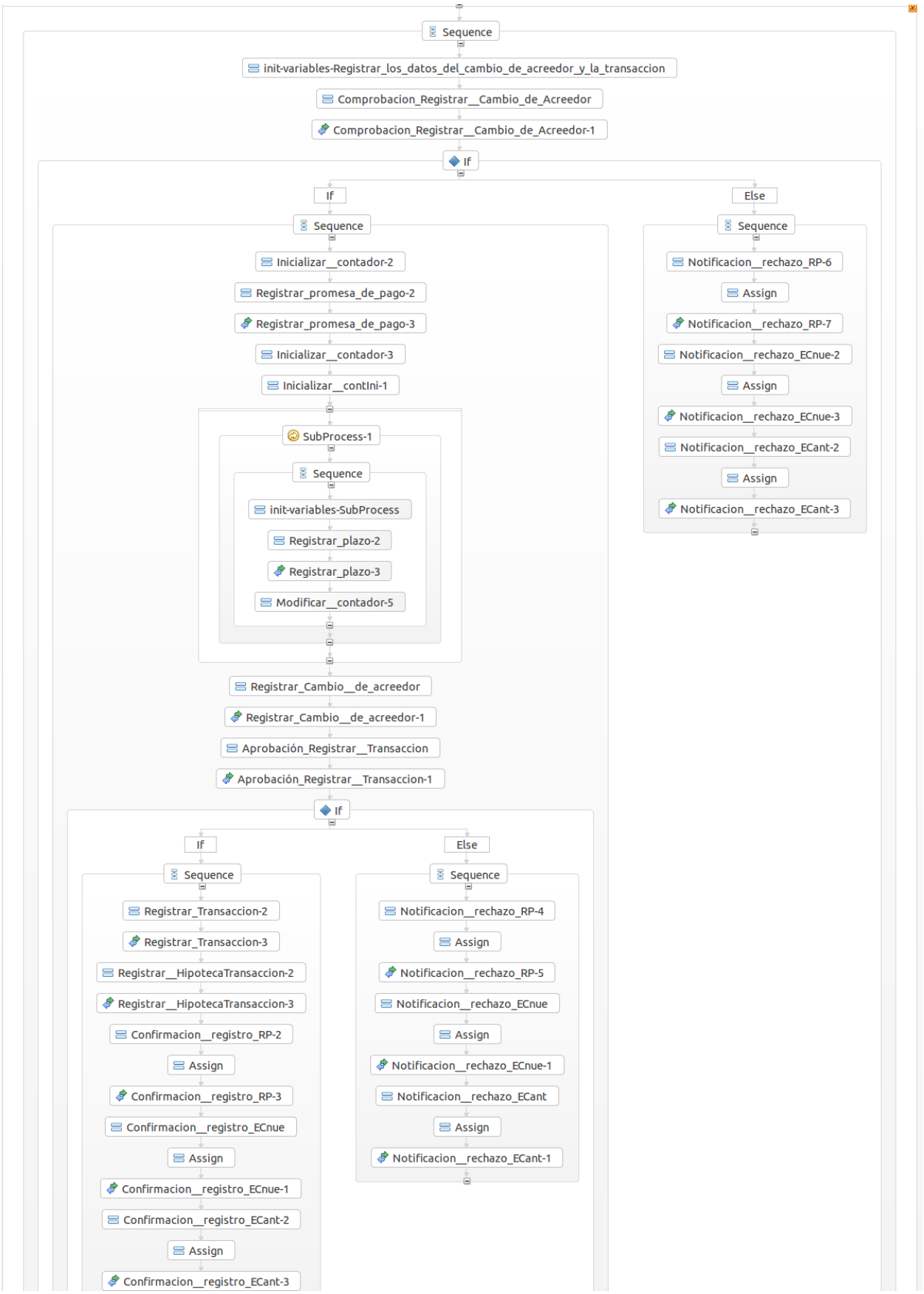


Figura 6.6.: Parte del código BPEL autogenerado por Intalio de Cambio de acreedor.

6.5.3. Variantes

Los siguientes requisitos se aplicarán para el proceso de Constitución de cédulas hipotecarias de registro: Proceso 3a Condiciones - Variante electrónica

- La entidad de crédito afectada está conectada al sistema.
- El notario afectado está conectado al sistema.
- La oficina del registro de la propiedad afectado está conectado al sistema.

6.5.4. Formularios

Para los procesos hay que utilizar los formularios: Mandato electrónico constitución cédula hipotecaria de registro, requisa electrónica creación de una cédula hipotecaria de registro y confirmación electrónica de la inscripción de la cédula hipotecaria de registro.

6.5.5. Comportamiento de la composición

A continuación, se procede a describir la estructura de la composición. El usuario de la entidad de crédito inicia el proceso.

- El sistema recibe mediante un mensaje el formulario con los datos para la constitución de la cédula hipotecaria de registro por parte de la entidad de crédito y se obtiene el código de activación. El sistema envía la orden al notario mediante un mensaje. El notario recibe el mensaje y rellena lo que falta en el formulario "Requisa electrónica creación de una cédula hipotecaria de registro". Se obtienen el código de transacción y los números de los participantes. Se introducen todos los documentos necesarios para el contrato de garantía en BD y se lo vuelve a enviar al sistema. El sistema le envía la requisa completa a la oficina del registro de la propiedad. El sistema comprobará si la inscripción de la hipoteca en la BD puede ser efectuada y si su contenido es correcto.

  Si es así, la hipoteca se actualiza en la base de datos y se registran los documentos en la base de datos.

- El sistema comprobará si la inscripción de la transacción en la BD puede ser efectuada y si su contenido es correcto.

  Si es así, el sistema registra la transacción en la BD.

- Si alguna de las dos comprobaciones anteriores es no, envía la notificación de rechazo a los 3 participantes del proceso. El sistema envía una notificación a los 3 participantes del proceso para la confirmación de la inscripción. Si se produce cualquier error durante el proceso, se captura y se notifica.

En la figura 6.7 se puede ver el diagrama BPMN del proceso Constitución de cédulas hipotecarias de registro, es una representación esquemática de lo que se ha hecho en Intalio. Se muestra en su nivel más alto nivel de abstracción, el flujo del proceso muestra sólo las actividades más importantes. Cada una de las actividades del nivel superior de abstracción se descomponen en flujos detallados. El proceso se va descomponiendo hasta que se alcanzan actividades atómicas (es decir, actividades que no se pueden descomponer aún más). En la figura 6.8 se puede observar una parte del comportamiento de este proceso en el editor de Eclipse en el nivel más bajo de abstracción, el código BPEL generado. Esta última figura, se refiere al subproceso "Obtener numero transaccion y numero de participante" y "insertar datos del documento" del diagrama BPMN.

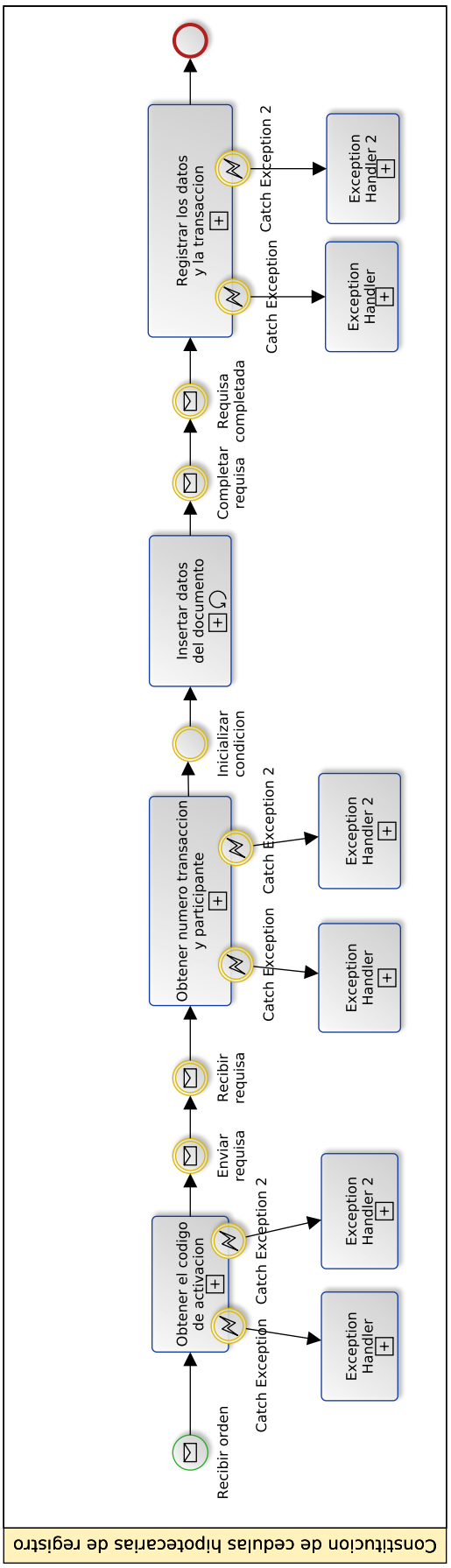


Figura 6.7.: Diagrama BPMN - Constitución de cédulas hipotecarias de registro.

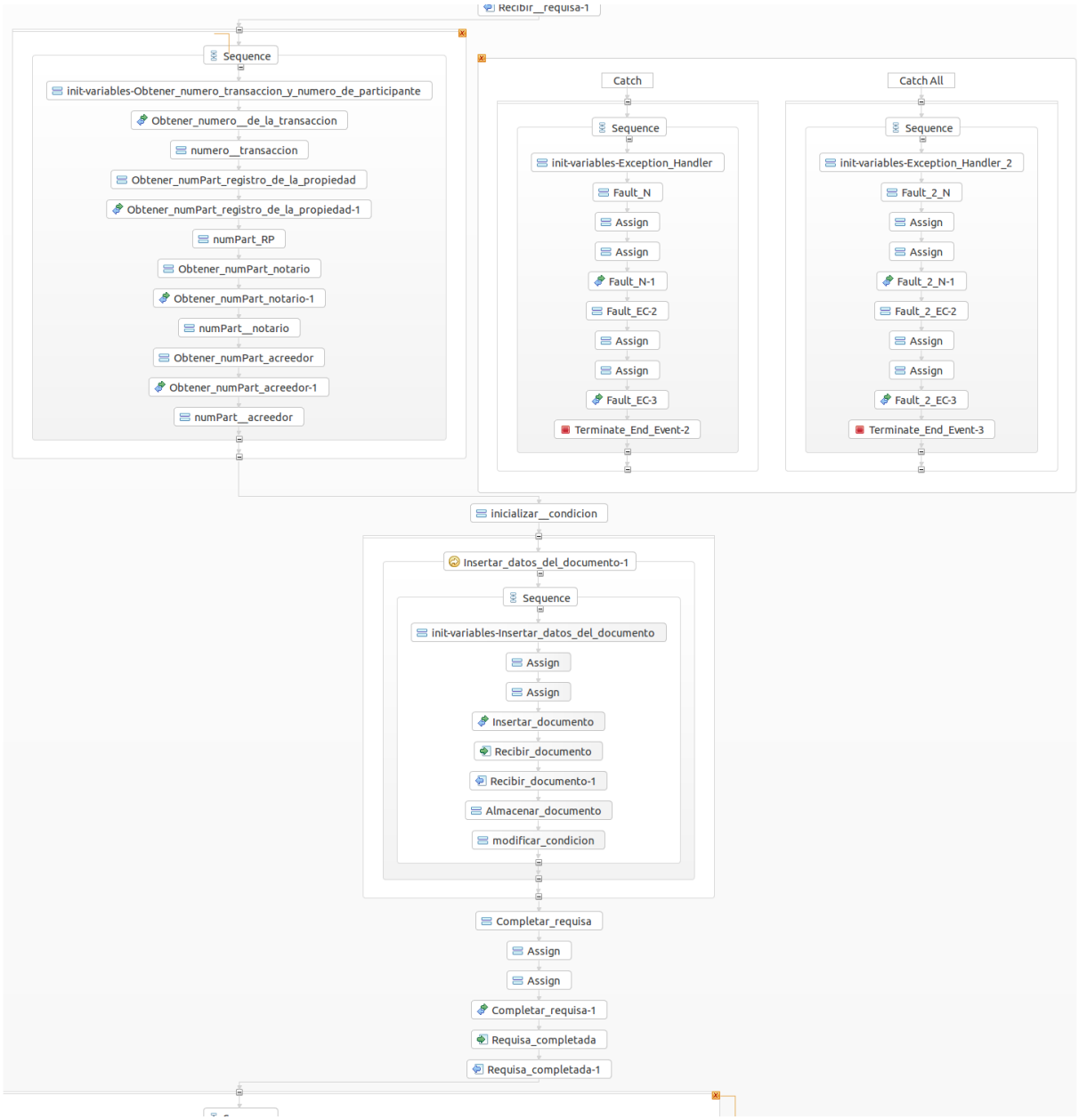


Figura 6.8.: Parte del código BPEL autogenerado por Intalio de Constitución de cédulas hipotecarias de registro.

7. Pruebas

En este capítulo se muestra la importancia de las pruebas para el software. Se comienza con una breve introducción, en la que se definen algunos conceptos básicos. Además, se detallan las distintas técnicas de diseño de casos de prueba y las estrategias de aplicación de las pruebas. Finalmente, se explican las diferentes pruebas que se han hecho para el desarrollo de este proyecto.

7.1. Introducción

Una de las características importantes del desarrollo de software basado en el ciclo de vida es la realización de controles periódicos. El objetivo de estos controles es evaluar la calidad de los productos generados para poder detectar posibles defectos cuanto antes. Las pruebas [47] constituyen un método más para poder validar y verificar el software. A continuación, se muestran las definiciones recogidas en el diccionario Institute of Electrical and Electronics Engineers (IEEE).

- **Prueba:** Una actividad en la cual un sistema o alguno de sus componentes se ejecuta en circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto.
- **Caso de prueba:** Un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular como, por ejemplo, probar un camino concreto de un programa o verifica el cumplimiento de un determinado requisito.
- **Defecto:** Un defecto en el software, como, por ejemplo, un proceso, una definición de datos o una tarea incorrecta en un programa.
- **Fallo:** La incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados.
- **Error:** La diferencia entre un valor calculado, observados o medido y el valor verdadero, especificado o teóricamente correcto.

7.2. El proceso de prueba

En la figura 7.1 se puede ver el proceso de prueba que comienza con la generación de un plan de pruebas en base a la documentación del proyecto y a la documentación del software a probar. A partir de dicho plan, se diseñan los casos de prueba, se ejecutan y los resultados obtenidos se comparan con los resultados esperados. Una vez evaluados los resultados de las pruebas, pueden realizarse dos actividades:

- Depurar los defectos.
- Analizar los errores.

La depuración puede corregir o no los defectos. Si no consigue localizarlos, puede ser necesario realizar pruebas adicionales para obtener más información Si se corrige un defecto, se debe volver probar el software para comprobar que el problema está resuelto. El análisis de errores puede servir para realizar predicciones de la fiabilidad del software y para detectar las causas más habituales de error y mejorar los procesos de desarrollo.

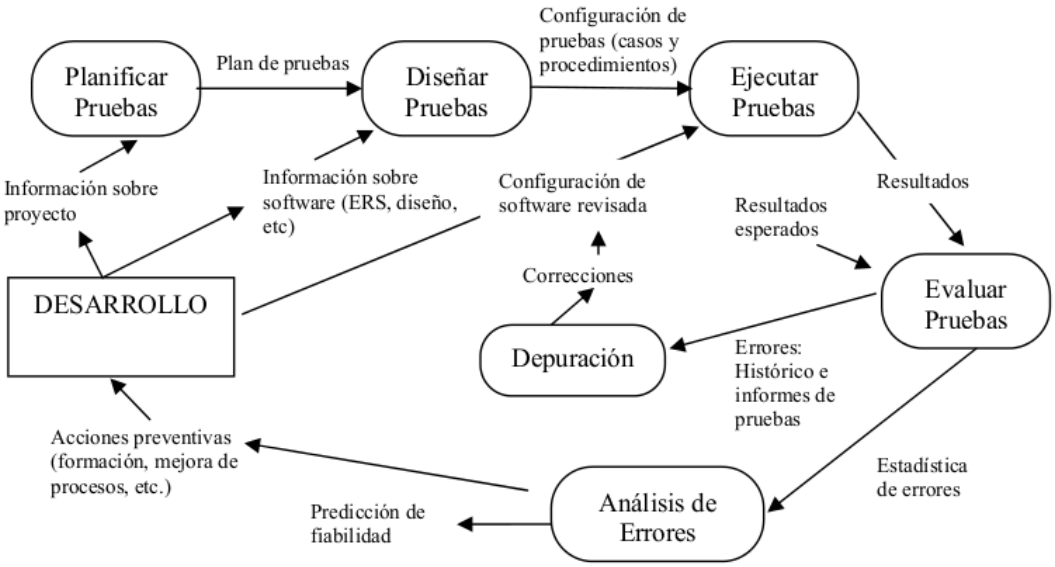


Figura 7.1.: Proceso de Prueba.

7.3. Técnicas de diseño de casos de prueba

La idea fundamental para el diseño de casos de prueba consiste en elegir aquellas posibilidades que, por sus características, se consideren representativas del resto. De esta forma se asume que, si no se detectan defectos en el software al ejecutar dichos casos, se puede tener cierto nivel de confianza en el programa no tiene defectos. Existen tres enfoques principales para el diseño de casos de prueba.

7.3.1. Enfoque estructural o de caja blanca

Consiste en centrarse en la estructura interna (implementación) del programa para elegir los casos de prueba. En este caso, la prueba exhaustiva del software consiste en probar todos los posibles caminos de ejecución que puedan trazarse. El diseño de los casos de prueba tiene que basarse en la elección de caminos importantes que ofrezcan cierta seguridad de que se van a descubrir los defectos.

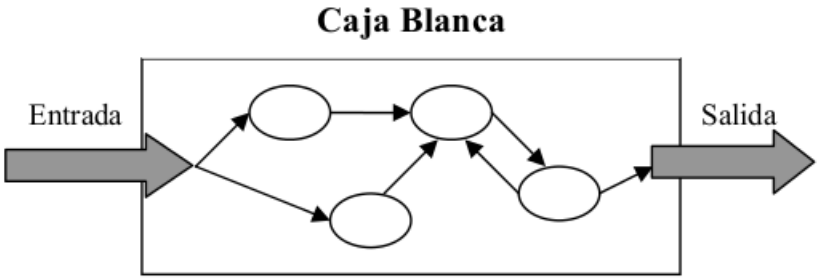


Figura 7.2.: Pruebas de caja blanca.

7.3.2. Enfoque funcional o de caja negra

Consiste en estudiar la especificación de las funciones, la entrada y la salida para definir los casos de prueba. En este caso, la prueba exhaustiva del software consiste en probar todas las posibles entradas y salidas del programa.

Las pruebas funcionales o de caja negra se centran en el estudio de la especificación de los requisitos del software, del análisis de las funciones que debe realizar, de las entradas y de las salidas.

Figura 7.3.: Pruebas de caja negra.

7.3.3. Enfoque aleatorio

Consiste en utilizar modelos (en muchas ocasiones estadísticos) que representan las posibles entradas al programa para crear a partir de ellos los casos de prueba. La prueba exhaustiva del software consiste en probar todas las posibles entradas al programa.

En las pruebas aleatorias se simulan las entradas de datos habituales del programa utilizando herramientas denominadas generadores de pruebas. La entrada de estas herramientas es una descripción de datos, las secuencias de entradas posibles y una estimación de la probabilidad que existe de que ocurran cada una de ellas en la vida real.

7.4. Estrategia de aplicación de las pruebas

La estrategia de aplicación y planificación de las pruebas tiene como finalidad crear distintos niveles de prueba con diferentes objetivos. En general, la estrategia de pruebas suele seguir una serie de etapas. En la figura 7.4 se muestra el modelo de ciclo de vida V. La idea principal es que las fases de desarrollo y pruebas corresponden a actividades de igual importancia. Las dos ramas de la V simbolizan esto.

Para cada nivel de especificación y la construcción, la rama derecha del modelo en V define un nivel de prueba correspondiente. La rama izquierda representa el proceso de desarrollo. Durante el desarrollo, el sistema está siendo diseñado gradualmente y finalmente implementado en un lenguaje de programación. La rama derecha representa el proceso de integración y las pruebas; con los elementos del programa, se van formando subsistemas más grandes (de integración), y su funcionalidad se prueba. Cuando se ha completado la prueba de aceptación de la totalidad del sistema se finaliza la integración y las pruebas.

7.4.1. Pruebas de unidad

Uno de los métodos más utilizados para realizar pruebas a nuestro software es el llamado de pruebas unitarias. Consiste en probar la lógica del módulo (caja blanca) y los distintos aspectos de las funciones que

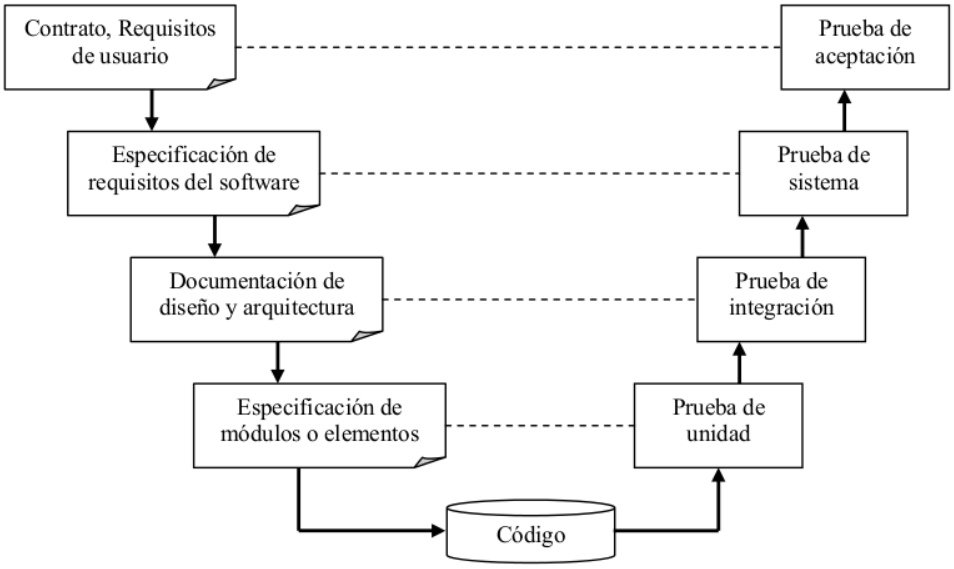


Figura 7.4.: Modelo de ciclo de vida V.

debe realizar el módulo (caja negra). La base de este método es el hacer pruebas en pequeños fragmentos de nuestro programa. Estos fragmentos deben ser unidades estructurales de nuestro programa encargados de una tarea específica. Tras realizar estas pruebas sobre los elementos unitarios del programa, se pueden eliminar gran parte de los errores de los que podría adolecer. Como ya se sabe cualquier prueba demuestra no la ausencia de errores sino que revela la presencia de ellos. El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores.

7.4.2. Pruebas de integración

Consiste en probar la integración de los módulos entre sí teniendo en cuenta la estructura modular del sistema y la interfaz intermodular, es decir, comprueba si los grupos de componentes interactúan de la manera especificada por el diseño técnico del sistema. No siempre es necesario terminar todas las pruebas de unidad antes de comenzar las pruebas de integración. Normalmente, estas dos pruebas se solapan. Hay dos tipos.

Integración incremental

Se integra el módulo que se va a probar con el conjunto de módulos ya probados. El número de módulos se incrementa progresivamente hasta formar el programa completo. En función del orden elegido para la integración, se diferencian dos tipos de integración: ascendente (se comienza por los módulos hoja) y descendente (se comienza por los módulos raíz).

Integración no incremental

Se prueba cada módulo por separado y, luego, se integran todos los módulos a la vez y se prueba el programa completo.

7.4.3. Pruebas del sistema

Consiste en probar la integración de todos los elementos del sistema, hardware, software y usuario:

1. Probar que el sistema cumple todos los requisitos funcionales considerando el sistema completo.

2. Comprobar el funcionamiento y rendimiento de las interfaces hardware, software, de usuario y de operador.
3. Probar la adecuación de la documentación de usuario.
4. Verificar la ejecución y rendimiento en condiciones límites y de sobrecarga.

7.4.4. Pruebas de aceptación

Antes de instalar y utilizar el software en la vida real, se a de ejecutar el último nivel de pruebas: la prueba de aceptación. Aquí, la atención se centra en la perspectiva del usuario y del cliente. La prueba de aceptación puede ser la única prueba de que los clientes están realmente involucrados o que pueden entender. Consiste en que el usuario compruebe si el producto final se ajusta a los requisitos especificados por él.

7.5. Plan de pruebas

Un plan de pruebas permite especificar lo que desea probar y cómo ejecutar dichas pruebas. La totalidad de las pruebas realizadas son pruebas de caja negra y blanca. Se recuerda que las pruebas de caja negra son aquellas que se centran en lo que se espera de un módulo, consiste en estudiar la especificación de las funciones, la entrada y la salida, al contrario que las pruebas de caja blanca que se centran en comprobar que la estructura interna del software es la adecuada.

El problema con las pruebas de caja negra es que el conjunto de los datos posibles a probar donde no se cumple la especificación es demasiado extenso. Para combatirlo, se deben seguir diferentes técnicas conocidas. La técnica que se utiliza para la capa de persistencia y de dominio es conocida como conjetura de errores, consiste en enumerar una lista de errores que suelen cometer los programadores o de situaciones que sean propensas a errores. A partir de esta lista se generan los casos de prueba. Esta técnica también se denomina generación de casos o valores especiales, ya que no se obtienen en base a otros métodos sino mediante la intuición o la experiencia.

7.5.1. Qué se va a probar

A continuación, se especifican las pruebas que se han considerado necesarias para cubrir la mayor parte de los aspectos de esta herramienta, se muestran agrupadas por módulos.

Entidades

En este módulo, se especifican las pruebas unitarias relacionadas con la capa de entidades perteneciente a la capa de dominio.

- **Crear objetos BO:** Para cada entidad se ha creado un «test» para comprobar que todos los objetos se creen correctamente. No se realizaron «test» para las entidades Sujeto y Entidad porque no hizo falta.

Servicios de dominio

En este otro módulo, se especifican tanto las pruebas unitarias como de integración relacionadas con la capa de servicio perteneciente a la capa de dominio. Las implementaciones de los servicios hacen uso del DAO, si se hacen test de la capa de servicio automáticamente se hacen pruebas de integración ya que el «framework» resuelve e instancia la dependencia servicio $\rightarrow$ DAO. A continuación, se muestran las pruebas realizadas a las típicas CRUD:

- **Introducir un objeto en la base de datos:** De forma genérica, para cada clase servicio se ha creado un «test» para comprobar que se almacena el objeto correctamente en la base de datos.

- **Borrar un objeto de la base de datos:** De forma genérica, para cada clase servicio se ha creado un «test» para comprobar que se borra correctamente de la base de datos.
- **Encontrar por una clave ya sea primaria o foránea un objeto en la base de datos:** De forma genérica, para cada clase servicio se ha creado un «test» para comprobar que se encuentra correctamente a través de su clave primaria o foránea en la base de datos.
- **Encontrar todos los objetos en la base de datos:** De forma genérica, para cada clase servicio se ha creado un «test» para comprobar que se encuentran todos los objetos introducidos anteriormente en la base de datos.
- **Actualizar un objeto de la base de datos:** De forma genérica, para cada clase servicio se ha creado un «test» para comprobar que se actualiza el objeto correctamente en la base de datos.
- **Borrar un objeto del que dependa otro:** En esta prueba, se va a intentar borrar un objeto de la base de datos que no puede ser borrado porque otro objeto depende de él y se produce una excepción controlada.

No se realizaron test para la clase servicio de la entidad Activacion al comprobarse en otra clase. En esta capa, además de las pruebas nombradas anteriormente, eran necesarias más pruebas para la validación de los atributos de la base de datos:

- **Introducir una fecha de firma mayor que el día actual o fecha de firma nula:** En estas pruebas, se introduce una fecha de firma mayor que el día actual o nula a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No puede firmarse después de ser registrada y tiene que firmarse.
- **Introducir una fecha inicial nula:** En esta prueba, se introduce una fecha inicial nula a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada porque al asignar el acreedor la fecha inicial nunca puede ser nula.
- **Introducir una fecha de constitución mayor que el día actual o nula:** En estas pruebas, se introduce una fecha de constitución mayor que el día actual o nula a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No puede constituirse después de ser registrada y para crearse debe tener esta fecha inicializada.
- **Introducir un IDE ya registrado o inválido ya sea por la expresión regular o por el dígito de control:** En estas pruebas, se introduce un IDE ya registrado o inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. Debe ser único.
- **Introducir un EREID ya registrado o inválido ya sea por la expresión regular o por el dígito de control:** En estas pruebas, se introduce un EREID inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada.
- **Introducir un EREID con los caracteres válidos como "/" o ".":** En estas pruebas, se introduce un EREID válido a un objeto para comprobar que se almacena en la base de datos correctamente.
- **Introducir un número del documento negativo:** En esta prueba, se introduce un número del documento negativo a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No puede ser negativo.
- **Introducir una tasa negativa o mayor que cien:** En estas pruebas, se introduce una tasa negativa o mayor que cien a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. La tasa está comprendida entre 0 y 100.

- **Introducir un importe inválido con coma, con más de dos decimales, demasiado grande o negativo:** En estas pruebas, se introduce importe inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. El importe debe tener dos decimales, separado de la parte entera con un punto, un máximo de 9 dígitos y no puede ser negativo.
- **Introducir un EGRID ya registrado o inválido ya sea por la expresión regular o por el dígito de control:** En estas pruebas, se introduce un EGRID ya registrado o inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. Debe ser único.
- **Introducir un número del catastro ya registrado o erróneo:** En estas pruebas, se introduce un número del catastro ya registrado o erróneo a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. Debe ser único.
- **Introducir una hoja de registro negativa:** En esta prueba, se introduce una hoja de registro negativa a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No puede ser negativa.
- **Introducir un código postal inválido ya sea por ser negativo, mayor que 9999 o menor que 1000 o por estar ya introducido:** En estas pruebas, se introduce un código postal inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. El código postal válido está entre el 1000 y el 9999.
- **Introducir un regNotariosCanton negativo:** En esta prueba, se introduce un regNotariosCanton negativo a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No puede ser negativo.
- **Introducir un código de activación nulo:** En esta prueba, se introduce un código de activación nulo a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No puede ser nulo.
- **Introducir una fecha de nacimiento incorrecta, menor de 18 años:** En esta prueba, se introduce una fecha de nacimiento incorrecta a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No se puede conceder una hipoteca a un menor de edad.
- **Introducir un NIP de persona física ya registrado o inválido con menos o más de 13 caracteres, negativo o incorrecto dígito de control:** En estas pruebas, se introduce un NIP ya registrado o inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. Debe ser único.
- **Introducir una comunidad nula:** En esta prueba, se introduce una comunidad nula a un objeto para comprobar que se almacena en la base de datos correctamente. Una persona física puede formar parte de una comunidad o no.
- **Introducir un sexo de más de un carácter o un carácter distinto a la expresión regular:** En esta prueba, se introduce un sexo inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada.
- **Introducir una clave primaria ya introducida:** En esta prueba, se introduce una clave primaria ya introducida a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. La clave primaria es única.
- **Introducir un iban inválido ya sea por la expresión regular o por el dígito de control:** En estas pruebas, se introduce un iban inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada.

- **Introducir un iban correcto pero de otro país:** En esta prueba, se introduce un iban correcto de España, por ejemplo, a un objeto para comprobar que se almacena en la base de datos correctamente.
- **Introducir una referencia incorrecta por la expresión regular:** En esta prueba, se introduce una referencia incorrecta a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada.
- **Introducir un teléfono inválido (negativo, más largo, más corto):** En estas pruebas, se introduce un teléfono inválido a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada.
- **Introducir un email incorrecto o ya registrado:** En estas pruebas, se introduce un email incorrecto o ya registrado a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. Debe ser único.
- **Introducir un username ya registrado:** En esta prueba, se introduce un username ya registrado a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. Debe ser único.
- **Introducir una promesa de pago nula en entidad de crédito:** En esta prueba, se introduce una promesa de pago nula a un objeto para comprobar que se almacena en la base de datos correctamente. Una entidad de crédito puede constituir una promesa de pago o no.
- **Introducir una transacción, un registro de la propiedad, una hipoteca, un lugar, un código de activación, una promesa de pago, un usuario, un DestOrden o un ExpOrden nulos:** En estas pruebas, se introduce una transacción, un registro de la propiedad, una hipoteca, un lugar, un código de activación, una promesa de pago, un usuario, un DestOrden o un ExpOrden nulos a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No pueden ser nulos.
- **Introducir una fecha válida nula:** En esta prueba, se introduce una fecha válida nula a un objeto para comprobar que no se almacena en la base de datos sino que se produce una excepción controlada. No puede ser nula.

Servicios web

En este otro módulo, se especifican tanto las pruebas unitarias como de integración relacionadas con la capa de servicios de entidad y de tarea pertenecientes a la capa API. Las implementaciones de los servicios web hacen uso de servicios de dominio de la capa de dominio, si se hacen test de los servicios web automáticamente se hacen pruebas de integración ya que el «framework» resuelve e instancia la dependencia servicio web → servicio de dominio. A continuación, se muestran las pruebas realizadas a los servicios web según la capa a la que pertenecen:

1. **La capa de los servicios de entidad.** Son necesarias las pruebas realizadas a las típicas CRUD:
   - **Introducir un objeto en la base de datos:** De forma genérica, para cada clase de servicio web se ha comprobado manualmente que se almacena el objeto correctamente en la base de datos.
   - **Borrar un objeto de la base de datos:** De forma genérica, para cada clase de servicio se ha comprobado manualmente que se borra correctamente de la base de datos.
   - **Encontrar por una clave ya sea primaria o foránea un objeto en la base de datos:** De forma genérica, para cada clase de servicio se ha comprobado manualmente que se encuentra correctamente a través de su clave primaria o foránea en la base de datos.

- **Encontrar todos los objetos en la base de datos:** De forma genérica, para cada clase de servicio se ha comprobado manualmente que se encuentran todos los objetos introducidos anteriormente en la base de datos.
- **Actualizar un objeto de la base de datos:** De forma genérica, para cada clase de servicio se ha comprobado manualmente que se actualiza el objeto correctamente en la base de datos.

2. **La capa de los servicios de tarea.** Son necesarias las siguientes pruebas:
   - **Registrar cambio de acreedor con datos válidos (entidades de crédito pertenecientes al sistema, hipoteca ya registrada en BD, acreedor antiguo asignado ya en BD, acreedor nuevo sin asignar):** Se comprueba que se registra correctamente el cambio de acreedor.
   - **Registrar cambio de acreedor con las mismas entidades de crédito:** Se comprueba que muestra una excepción.
   - **Registrar cambio de acreedor con EREID vacío porque los demás se obliga a rellenarlos:** Se comprueba que muestra una excepción.
   - **Registrar cambio de acreedor con antiguo acreedor no existe en BD:** Se comprueba que muestra una excepción.
   - **Registrar cambio de acreedor con antiguo acreedor existe en BD pero no es el último antiguo acreedor:** Se comprueba que es uno de los antiguos acreedores pero como fchFin no está vacía no es el último antiguo acreedor y muestra una excepción.
   - **Registrar cambio de acreedor con nuevo acreedor creado ya en BD:** Se comprueba que muestra una excepción.
   - **Registrar cambio de acreedor con hipoteca no registrada:** Se comprueba que muestra una excepción.
   - **Registrar cambio de acreedor con entidad de crédito (nueva) no perteneciente al sistema:** Se comprueba que muestra una excepción.
   - **Registrar cambio de acreedor con entidad de crédito (antigua) no perteneciente al sistema:** Se comprueba que muestra una excepción.
   - **Validar inmueble con los parámetros vacíos:** Se comprueba que devuelve falso.
   - **Validar inmueble con los dos parámetros:** Se comprueba que devuelve falso.
   - **Validar inmueble con EGRID válido, no registrado y el número del inmueble vacío:** Se comprueba que no exista registrado en la BD. Devuelve verdadero.
   - **Validar inmueble con EGRID no registrado pero de tamaño incorrecto y el número del inmueble vacío:** Se comprueba que devuelve falso.
   - **Validar inmueble con EGRID no registrado pero de «pattern» incorrecto y el número del inmueble vacío:** Se comprueba que devuelve falso.
   - **Validar inmueble con EGRID registrado y el número del inmueble vacío:** Se comprueba que devuelve falso.
   - **Validar inmueble con EGRID vacío, el número del inmueble válido y no registrado:** Se comprueba que no exista registrado en la BD. Devuelve verdadero.
   - **Validar inmueble con EGRID vacío, el número del inmueble no registrado y siendo una cadena de caracteres:** Se comprueba que devuelve falso.
   - **Validar inmueble con EGRID vacío, el número del inmueble no registrado y siendo un número negativo:** Se comprueba que devuelve falso.

- **Validar inmueble con EGRID vacío, el número del inmueble no registrado y siendo un número decimal:** Se comprueba que devuelve falso.
- **Validar inmueble con EGRID vacío y el número del inmueble registrado:** Se comprueba que devuelve falso.
- **Inmueble en el mismo distrito que el registro de la propiedad con los parámetros vacíos:** Se comprueba que devuelve falso.
- **Inmueble en el mismo distrito que el registro de la propiedad con municipioBarrio válido y el ide vacío:** Se comprueba que devuelve falso.
- **Inmueble en el mismo distrito que el registro de la propiedad con municipioBarrio vacío y el ide válido:** Se comprueba que devuelve falso.
- **Inmueble en el mismo distrito que el registro de la propiedad con municipioBarrio válido y el ide válido, están en el mismo distrito:** Se comprueba que el ide pertenece a un registro de la propiedad registrado en la BD y que está en el mismo distrito que el distrito contenido en el municipioBarrio del inmueble. Devuelve verdadero.
- **Inmueble en el mismo distrito que el registro de la propiedad con municipioBarrio válido y el ide válido, no están en el mismo distrito:** Se comprueba que el ide pertenece a un registro de la propiedad registrado en la BD y que no está en el mismo distrito que el distrito contenido en el municipioBarrio del inmueble. Devuelve falso.
- **Inmueble en el mismo distrito que el registro de la propiedad con municipioBarrio válido y el ide no registrado:** Se comprueba que devuelve falso. El ide no corresponde a ningún registro de la propiedad registrado en la BD. Devuelve falso.
- **Aprobación registrar hipoteca sin introducir nada:** Devuelve falso.
- **Aprobación registrar hipoteca con ereid vacío:** Devuelve falso.
- **Aprobación registrar hipoteca con ereid mayor que 22:** Devuelve falso.
- **Aprobación registrar hipoteca con ereid registrado:** Hipoteca registrada en BD. Devuelve falso.
- **Aprobación registrar hipoteca con ereid no registrado e incorrecto por «pattern»:** Devuelve falso.
- **Aprobación registrar hipoteca con ereid no registrado e incorrecto por dígitos de control:** Devuelve falso.
- **Aprobación registrar hipoteca con ereid válido y no registrado:** Se comprueba que no exista registrado en la BD. Devuelve verdadero.
- **Aprobación registrar transacción sin introducir nada:** Devuelve falso.
- **Aprobación registrar transacción con número de la transacción negativo:** Devuelve falso.
- **Aprobación registrar transacción con número de la transacción registrado:** Transacción registrada en BD. Devuelve falso.
- **Aprobación registrar transacción con número de la transacción no registrado y positivo:** Se comprueba que no exista registrado en la BD. Devuelve verdadero.
- **Comprobar que la entidad de crédito es la que financiaba hasta el momento con parámetros vacíos:** Devuelve falso.
- **Comprobar que la entidad de crédito es la que financiaba hasta el momento con algún parámetro vacío:** Devuelve falso.

- **Comprobar que la entidad de crédito es la que financiaba hasta el momento con ereid no registrado en BD:** Se comprueba que no exista hipoteca registrada en la BD. Devuelve falso.
- **Comprobar que la entidad de crédito es la que financiaba hasta el momento con numPart no registrado en BD:** Se comprueba que no exista entidad de crédito registrado en la BD. Devuelve falso.
- **Comprobar que la entidad de crédito es la que financiaba hasta el momento con numPart y ereid registrado en BD y con fchFin vacío:** Se comprueba que el numPart pertenece a una entidad de crédito registrada en la BD, ereid a una hipoteca también registrada y que la fchFin está vacía y por lo tanto, era el que financiaba hasta el momento. Devuelve verdadero.
- **Comprobar que la entidad de crédito es la que financiaba hasta el momento con numPart y ereid registrado en BD y con fchFin:** Se comprueba que el numPart pertenece a una entidad de crédito registrada en la BD, ereid a una hipoteca también registrada y que la fchFin no está vacía y por lo tanto, no era el que financiaba hasta el momento. Devuelve falso.
- **Aprobación del nuevo acreedor con parámetro ereid vacío:** Se comprueba que devuelve falso.
- **Aprobación del nuevo acreedor con numPart y ereid registrados:** Se comprueba que el acreedor pertenece al sistema, la hipoteca está registrada y devuelve verdadero. Se puede registrar como nuevo acreedor.
- **Aprobación del nuevo acreedor con numPart no registrado y ereid registrado:** Se comprueba que el acreedor no pertenece al sistema y devuelve falso.
- **Aprobación del nuevo acreedor con numPart registrado y ereid no registrado:** Se comprueba que el acreedor pertenece al sistema pero la hipoteca no existe en BD y devuelve falso.
- **Comprobar que las entidades de crédito tienen el mismo registro de la propiedad con parámetros vacíos:** Devuelve falso.
- **Comprobar que las entidades de crédito tienen el mismo registro de la propiedad con algún parámetro vacío:** Devuelve falso.
- **Comprobar que las entidades de crédito tienen el mismo registro de la propiedad con ideNueva correcta e ideAtigua incorrecta:** No existe la entidad de crédito antigua en la BD. Devuelve falso.
- **Comprobar que las entidades de crédito tienen el mismo registro de la propiedad con ideAtigua correcta e ideNueva incorrecta:** No existe la entidad de crédito nueva en la BD. Devuelve falso.
- **Comprobar que las entidades de crédito tienen el mismo registro de la propiedad con ideAtigua correcta, ideNueva correcta y no tienen el mismo registro de propiedad:** Se comprueba que las dos entidades de crédito están registradas en BD y no tienen el mismo registro de la propiedad. Devuelve falso.
- **Comprobar que las entidades de crédito tienen el mismo registro de la propiedad con ideAtigua correcta, ideNueva correcta y tienen el mismo registro de propiedad:** Se comprueba que las dos entidades de crédito están registradas en BD y tienen el mismo registro de la propiedad. Devuelve verdadero.
- **Comprobar que los plazos tienen una fecha válida de como máximo 18 meses con parámetro vacío o nulo:** Devuelve falso.
- **Comprobar que los plazos tienen una fecha válida de como máximo 18 meses con parámetro una cadena con una fecha:** Devuelve verdadero.

- **Comprobar que los plazos tienen una fecha válida de como máximo 18 meses con parámetro una cadena con más de dos fechas desordenadas y la diferencia entre fechas es menor o igual a 18 meses:** Devuelve verdadero.
- **Comprobar que los plazos tienen una fecha válida de como máximo 18 meses con parámetro una cadena con más de dos fechas ordenadas y la diferencia entre fechas es menor o igual a 18 meses:** Devuelve verdadero.
- **Comprobar que los plazos tienen una fecha válida de como máximo 18 meses con parámetro una cadena con más de dos fechas desordenadas y la diferencia entre fechas es mayor a 18 meses:** Devuelve falso.
- **Comprobar que los plazos tienen una fecha válida de como máximo 18 meses con parámetro una cadena con más de dos fechas ordenadas y la diferencia entre fechas es mayor a 18 meses:** Devuelve falso.
- **Hipoteca registrada con ereid no registrado:** Devuelve falso.
- **Hipoteca registrada con ereid registrado:** Se comprueba que exista registrado en la BD. Devuelve verdadero.
- **Obtener el número del participante introduciendo un ide válido:** Se comprueba que devuelve el número del participante correspondiente a dicho ide.
- **Obtener el número del participante introduciendo un ide inválido:** Se comprueba que devuelve una excepción.
- **Obtener el número del usuario introduciendo un email válido:** Se comprueba que devuelve el número del usuario correspondiente a dicho email.
- **Obtener el número del usuario introduciendo un email inválido:** Se comprueba que devuelve una excepción.
- **Obtener el número de la transacción:** Se comprueba que devuelve el número de la transacción siguiente al último registrado en la BD.
- **Obtener el código de activación introduciendo un ide válido:** Se comprueba que devuelve el código de activación correspondiente a dicho ide.
- **Obtener el código de activación introduciendo un ide inválido:** Se comprueba que devuelve una excepción.
- **Obtener el número de la promesa de pago:** Se comprueba que devuelve el id de la promesa de pago siguiente a la última registrada en la BD.

Composiciones

Para terminar y asegurar el correcto funcionamiento del proyecto, se especifican las pruebas unitarias para la capa de composiciones. A continuación, se muestran las pruebas necesarias realizadas según la composición:

1. **Registrar cédula hipotecaria:**
   - **RegistrarCedula1Test.** Registrar cédula hipotecaria con variante del acuerdo previo, seleccionando persona física en la declaración del acuerdo y continuar con el proceso, seleccionando EGRID, siendo EGRID correcto y no registrado, seleccionar continuar con el proceso, seleccionando tipo de propietario persona física, con inmueble y registro de la propiedad que están en el mismo distrito, propietario continuar con el proceso, referencia continuar con el proceso, con Obtener el número del usuario introduciendo un email válido, con Obtener el número del participante introduciendo un ide válido, con hipoteca no registrada y EREID correcto, con

contenido Hipoteca correcto, con contenido AsignarAcreedor correcto, con contenido SujetoHipoteca correcto, con contenido Inmueble correcto, con contenido Referencia correcto, con contenido Firmante correcto, con transacción no registrada, con contenido Transaccion correcto y con contenido HipotecaTransaccion correcto: Se comprueba que se realiza el registro correctamente en la BD.

- **RegistrarCedula2Test.** Registrar cédula hipotecaria con variante del acuerdo directo, seleccionando EGRID, siendo EGRID correcto y no registrado, seleccionar continuar con el proceso, seleccionando tipo de propietario persona física, con inmueble y registro de la propiedad que están en el mismo distrito, propietario continuar con el proceso, referencia continuar con el proceso, con Obtener el número del usuario introduciendo un email válido, con Obtener el número del participante introduciendo un ide válido, seleccionando persona física en la declaración del acuerdo y continuar con el proceso, con hipoteca no registrada y EREID correcto, con contenido Hipoteca correcto, con contenido AsignarAcreedor correcto, con contenido SujetoHipoteca correcto, con contenido Inmueble correcto, con contenido Referencia correcto, con contenido Firmante correcto, con transacción no registrada, con contenido Transaccion correcto y con contenido HipotecaTransaccion correcto: Se comprueba que se realiza el registro correctamente en la BD.
- **PrevioComunidadTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero seleccionando tipo de propietario comunidad.
- **DirectoComunidadTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula2Test pero seleccionando tipo de propietario comunidad.
- **PrevioPersonaJuridicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero seleccionando tipo de propietario persona jurídica.
- **DirectoPersonaJuridicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula2Test pero seleccionando tipo de propietario persona jurídica.
- **NotificacionPapelTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero sin notificar electrónicamente a la entidad de crédito.
- **PrevioVariasPersonaFisicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero seleccionando persona física 2 veces en la declaración del acuerdo y seleccionar de nuevo tipo propietario persona física 2 veces.
- **DirectoVariasPersonaFisicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula2Test pero seleccionando persona física 2 veces en la declaración del acuerdo y seleccionar de nuevo tipo propietario persona física 2 veces.
- **PrevioVariasPersonaJuridicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero seleccionando persona jurídica 2 veces en la declaración del acuerdo y seleccionar de nuevo tipo propietario persona jurídica 2 veces.
- **DirectoVariasPersonaJuridicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula2Test pero seleccionando persona jurídica 2 veces en la declaración del acuerdo y seleccionar de nuevo tipo propietario persona jurídica 2 veces.
- **PrevioVariosInmueblesTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero introduciendo 2 inmuebles.
- **DirectoVariosInmueblesTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula2Test pero introduciendo 2 inmuebles.
- **PrevioNumInmuebleTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero seleccionando numInmueble, siendo numInmueble correcto y no registrado.

- **DirectoNumInmuebleTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula2Test pero seleccionando numInmueble, siendo numInmueble correcto y no registrado.
- **EGRIDIncorrectoTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con EGRID incorrecto y no registrado, seleccionar continuar con el proceso: Se comprueba que el EGRID es incorrecto y no se puede seguir con el proceso, se finaliza el proceso y se envía una notificación.
- **ValidarEGRIDFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **NumInmuebleIncorrectoTest.** Este caso de prueba sigue un comportamiento semejante al que se define en PrevioNumInmuebleTest pero siendo numInmueble incorrecto y no registrado, seleccionar continuar con el proceso: Se comprueba que numInmueble es incorrecto y no se puede seguir con el proceso, se finaliza el proceso y se envía una notificación.
- **ValidarNumInmuebleFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **DistintoDistritoTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero inmueble y registro de la propiedad no están en el mismo distrito: Se comprueba que no pertenecen al mismo distrito, se finaliza el proceso y se envía una notificación.
- **ValidarMismoDistritoFaultCatchAllTest.** Igual que el anterior pero se comprueba que los datos son incorrectos y se lanza una excepción.
- **ObtenerNumUsuarioFaultCathAllTest y ObtenerNumUsuarioFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con Obtener el número del usuario introduciendo un email inválido o cualquier otro error: Se comprueba que devuelve una excepción.
- **ObtenerNumTransacFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero se comprueba que devuelve una excepción al obtener el número de la transacción.
- **ObtenerNumPartFaultCathAllTest y ObtenerNumPartFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con Obtener el número del participante introduciendo un ide inválido o cualquier otro error: Se comprueba que devuelve una excepción.
- **HipotecaExisteTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con hipoteca registrada: Se comprueba que la hipoteca ya existe, se finaliza el proceso y se envía una notificación.
- **HipotecaExisteFaultCathAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **HipotecaFaultCathAllTest y HipotecaFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con contenido Hipoteca incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **AsignarAcreedorFaultCathAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con contenido AsignarAcreedor incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **SujetoHipotecaFaultCathAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con contenido SujetoHipoteca incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.

- **InmuebleFaultCathAllTest y InmuebleFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con contenido Inmueble incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **ReferenciaFaultCathAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con contenido Referencia incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **FirmanteFaultCathAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con contenido Firmante incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **TransaccionExisteTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con transacción registrada: Se comprueba que la transacción ya existe, se finaliza el proceso y se envía una notificación.
- **TransaccionExisteFaultCathAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **TransaccionFaultCathAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con contenido Transaccion incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **HipotecaTransaccionFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en RegistrarCedula1Test pero con contenido HipotecaTransaccion incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.

2. **Cambio de acreedor:**
   - **CambioAcreedor1Test.** Registrar cédula hipotecaria con variante del acuerdo previo, seleccionando persona física en la declaración del acuerdo y continuar con el proceso, seleccionando EGRID, siendo EGRID correcto y no registrado, seleccionar continuar con el proceso, seleccionando tipo de propietario persona física, con inmueble y registro de la propiedad que están en el mismo distrito, con las entidades de crédito que tienen el mismo registro de la propiedad, propietario continuar con el proceso, referencia continuar con el proceso, con Obtener el número del usuario introduciendo un email válido, con Obtener el número del participante introduciendo un ide válido, aceptar la promesa, con la fecha válida de los plazos de como máximo 18 meses, con hipoteca no registrada y EREID correcto, con contenido Hipoteca correcto, con contenido SujetoHipoteca correcto, con contenido Inmueble correcto, con contenido Referencia correcto, con contenido Firmante correcto, con contenido AsignarAcreedor (nuevo) correcto, con contenido AsignarAcreedor (antiguo) correcto, con contenido PromesaPago correcto, con contenido Plazo correcto, con transacción no registrada, con contenido Transaccion correcto y con contenido HipotecaTransaccion correcto: Se comprueba que se realiza el registro correctamente en la BD.
   - **CambioAcreedor2Test.** Registrar cédula hipotecaria con variante del acuerdo directo, seleccionando persona física en la declaración del acuerdo y continuar con el proceso, seleccionando EGRID, siendo EGRID correcto y no registrado, seleccionar continuar con el proceso, seleccionando tipo de propietario persona física, con inmueble y registro de la propiedad que están en el mismo distrito, con las entidades de crédito que tienen el mismo registro de la propiedad, propietario continuar con el proceso, referencia continuar con el proceso, con Obtener el número del usuario introduciendo un email válido, con Obtener el número del participante introduciendo un ide válido, aceptar la promesa, con la fecha válida de los plazos de como máximo 18 meses, con hipoteca no registrada y EREID correcto, con contenido Hipoteca correcto, con contenido SujetoHipoteca correcto, con contenido Inmueble correcto, con contenido Referencia correcto,

con contenido Firmante correcto, con contenido AsignarAcreedor (nuevo) correcto, con contenido AsignarAcreedor (antiguo) correcto, con contenido PromesaPago correcto, con contenido Plazo correcto, con transacción no registrada, con contenido Transaccion correcto y con contenido HipotecaTransaccion correcto: Se comprueba que se realiza el registro correctamente en la BD.

- **NotificacionPapelTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero sin notificar electrónicamente a la entidad de crédito.
- **PrevioPersonaJuridicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero seleccionando tipo de propietario persona jurídica.
- **DirectoPersonaJuridicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor2Test pero seleccionando tipo de propietario persona jurídica.
- **PrevioComunidadTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero seleccionando tipo de propietario comunidad.
- **DirectoComunidadTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor2Test pero seleccionando tipo de propietario comunidad.
- **VariasPromesasDePagoTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero rechazando promesa teniendo que volver a repetir una nueva promesa y aceptarla.
- **PrevioVariasPersonaFisica.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero seleccionando persona física 2 veces en la declaración del acuerdo y seleccionar de nuevo tipo propietario persona física 2 veces.
- **DirectoVariasPersonaFisica.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor2Test pero seleccionando persona física 2 veces en la declaración del acuerdo y seleccionar de nuevo tipo propietario persona física 2 veces.
- **PrevioVariasPersonaJuridicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero seleccionando persona jurídica 2 veces en la declaración del acuerdo y seleccionar de nuevo tipo propietario persona jurídica 2 veces.
- **DirectoVariasPersonaJuridicaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor2Test pero seleccionando persona jurídica 2 veces en la declaración del acuerdo y seleccionar de nuevo tipo propietario persona jurídica 2 veces.
- **PrevioNumInmuebleTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero seleccionando numInmueble, siendo numInmueble correcto y no registrado.
- **DirectoNumInmuebleTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor2Test pero seleccionando numInmueble, siendo numInmueble correcto y no registrado.
- **PrevioVariosInmuebleTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero introduciendo 2 inmuebles.
- **DirectoVariosInmuebleTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor2Test pero introduciendo 2 inmuebles.
- **EGRIDIncorrectoTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con EGRID incorrecto y no registrado, seleccionar continuar con el proceso: Se comprueba que el EGRID es incorrecto y no se puede seguir con el proceso, se finaliza el proceso y se envía una notificación.

- **ValidarEGRIDFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **NumInmuebleIncorrectoTest.** Este caso de prueba sigue un comportamiento semejante al que se define en PrevioNumInmuebleTest pero siendo numInmueble incorrecto y no registrado, seleccionar continuar con el proceso: Se comprueba que numInmueble es incorrecto y no se puede seguir con el proceso, se finaliza el proceso y se envía una notificación.
- **ValidarNumInmuebleFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **DistintoDistritoTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero inmueble y registro de la propiedad no están en el mismo distrito: Se comprueba que no pertenecen al mismo distrito, se finaliza el proceso y se envía una notificación.
- **ValidarMismoDistritoFaultCatchAllTest.** Igual que el anterior pero se comprueba que los datos son incorrectos y se lanza una excepción.
- **DistintoRegistroPropiedadTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con las entidades de crédito no tienen el mismo registro de la propiedad: Se comprueba que no tienen el mismo registro de la propiedad, se finaliza el proceso y se envía una notificación.
- **ValidarMismoRegistroPropiedadFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **ObtenerNumUsuarioFaultCatchAllTest y ObtenerNumUsuarioFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con Obtener el número del usuario introduciendo un email inválido: Se comprueba que devuelve una excepción.
- **FechaPlazoNoValidaTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con la fecha válida de los plazos de más de 18 meses: Se comprueba que entre fecha y fecha se supera los 18 meses, se finaliza el proceso y se envía una notificación.
- **ValidarFechaPlazosFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **ObtenerNumTransacFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero se comprueba que devuelve una excepción al obtener el número de la transacción.
- **ObtenerNumPromesaFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero se comprueba que devuelve una excepción al obtener el número de la promesa.
- **ObtenerNumPartFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con Obtener el número del participante introduciendo un ide inválido o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **HipotecaExisteTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con hipoteca registrada: Se comprueba que la hipoteca ya existe, se finaliza el proceso y se envía una notificación.
- **AprobacionRegistrarHipotecaFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.

- **HipotecaFaultCatchAllTest y HipotecaFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido Hipoteca incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **SujetoHipotecaFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido SujetoHipoteca incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **InmuebleFaultCatchAllTest y InmuebleFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido Inmueble incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **ReferenciaFaultCacthAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido Referencia incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **FirmanteFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido Firmante incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **PromesaPagoFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido PromesaPago incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **PlazoFaultCatchAllTest y PlazoFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido Plazo incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **AsignarAcreedorFaultCatchAllTest y AsignarAcreedorFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido AsignarAcreedor incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **TransaccionExisteTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con transacción registrada: Se comprueba que la transacción ya existe, se finaliza el proceso y se envía una notificación.
- **AprobacionRegistrarTransaccionFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **TransaccionFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido Transaccion incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **HipotecaTransaccionFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor1Test pero con contenido HipotecaTransaccion incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **CambioAcreedor3Test.** Cambio de acreedor con cédula hipotecaria de registro ya disponible, con aprobación del antiguo acreedor válido, con las entidades de crédito que tienen el mismo registro de la propiedad, con Obtener el número del usuario introduciendo un email válido, con Obtener el número del participante introduciendo un ide válido, seleccionando continuar con el proceso, aceptar la promesa, con la fecha válida de los plazos de como máximo 18 meses, con hipoteca registrada, con contenido AsignarAcreedor (nuevo) correcto, con contenido AsignarAcreedor (antiguo) correcto, con contenido PromesaPago correcto, con contenido Plazo correcto, con transacción no registrada, con contenido Transaccion correcto y con contenido HipotecaTransaccion correcto: Se comprueba que se realiza el registro correctamente en la BD.

- **VariasPromesasDePagoTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero rechazando promesa teniendo que volver a repetir una nueva promesa y aceptarla.
- **AntiguoAcreedorNoValidoTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con aprobación del antiguo acreedor inválido: Se comprueba que no es el mismo acreedor registrado o no se ha registrado, se finaliza el proceso y se envía una notificación.
- **ValidarEntidadCreditoAntiguaFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **DistintoRegistroPropiedadTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con las entidades de crédito no tienen el mismo registro de la propiedad: Se comprueba que no tienen el mismo registro de la propiedad, se finaliza el proceso y se envía una notificación.
- **ValidarMismoRegistroPropiedadFaultCatchAllTest2.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **FechaPlazoNoValidaTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con la fecha válida de los plazos de más de 18 meses: Se comprueba que entre fecha y fecha se supera los 18 meses, se finaliza el proceso y se envía una notificación.
- **ValidarFechaPlazosFaultCatchAllTest2.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **ObtenerNumTransacFaultCatchAllTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero se comprueba que devuelve una excepción al obtener el número de la transacción.
- **ObtenerNumPromesaFaultCatchAllTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero se comprueba que devuelve una excepción al obtener el número de la promesa.
- **ObtenerNumPartFaultCatchAllTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con Obtener el número del participante introduciendo un ide inválido o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **NoAprobadoRegistrarNuevoAcreedorTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con hipoteca no registrada: Se comprueba que la hipoteca no existe, se finaliza el proceso y se envía una notificación.
- **RegistrarCambioAcreedorFaultTest y RegistrarCambioAcreedorFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con contenido AsignarAcreedor (nuevo y antiguo) incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
- **PromesaPagoFaultCatchAllTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con contenido PromesaPago incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **PlazoFaultCatchAllTest2 y PlazoFaultTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con contenido Plazo incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.

- **TransaccionExisteTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con transacción registrada: Se comprueba que la transacción ya existe, se finaliza el proceso y se envía una notificación.
- **AprobacionRegistrarTransaccionFaultCatchAllTest2.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **TransaccionFaultCatchAllTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero pero con contenido Transaccion incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **HipotecaTransaccionFaultCatchAllTest2.** Este caso de prueba sigue un comportamiento semejante al que se define en CambioAcreedor3 pero con contenido HipotecaTransaccion incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.

3. **Constitución de cédulas hipotecarias de registro:**
   - **ConstitucionCedula1Test.** Constitución de cédulas hipotecarias de registro con ide válido para obtener el código de activación, con ide válido para obtener el número del participante, seleccionando continuar proceso, con hipoteca registrada, con actualizar Hipoteca contenido correcto, con transaccion no registrada, con contenido Transaccion correcto, con contenido HipotecaTransaccion correcto y con contenido Documento correcto: Se comprueba que se realiza el registro correctamente en la BD.
   - **ConstitucionCedula2Test.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero se introducen 2 documentos.
   - **ObtenerCodActivacionFaultCatchAllTest y ObtenerCodActivacionFaultTest.** Constitución de cédulas hipotecarias de registro con ide inválido para obtener el código de activación o cualquier otro error: Se comprueba que devuelve una excepción.
   - **ObtenerNumTransacFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero se comprueba que devuelve una excepción al obtener el número de la transacción.
   - **ObtenerNumPartFaultCatchAllTest y ObtenerNumPartFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero con ide inválido para obtener el número del participante: Se comprueba que devuelve una excepción.
   - **HipotecaNoExisteTest.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero con hipoteca no registrada: Se comprueba que la hipoteca no está registrada y no se puede seguir con el proceso, se finaliza el proceso y se envía una notificación.
   - **HipotecaNoExisteFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
   - **UpdateHipotecaFaultCatchAllTest y UpdateHipotecaFaultTest.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero con actualizar Hipoteca contenido incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.
   - **TransaccionExisteTest.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero con transaccion registrada: Se comprueba que la transacción ya existe, se finaliza el proceso y se envía una notificación.
   - **TransaccionExisteFaultCatchAllTest.** Igual que el anterior pero captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.

- **TransaccionFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero con contenido Transaccion incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **HipotecaTransaccionFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero con contenido HipotecaTransaccion incorrecto o captura cualquier error, por ejemplo, sin conexión a la BD y se lanza una excepción.
- **DocumentoFaultCatchAllTest.** Este caso de prueba sigue un comportamiento semejante al que se define en ConstitucionCedula1Test pero con contenido Documento incorrecto: Se comprueba que los datos son incorrectos y se lanza una excepción.

7.5.2. Cómo se va a probar

Como se comentó en capítulos anteriores, se decidió usar Spring y JPA (Hibernate) para capa de persistencia y de dominio. Para las pruebas se ha usado JUnit y también, DbUnit para llenar una base de datos. Para la capa de los servicios web, las pruebas se han realizado manualmente, ejecutando cada servicio web e introduciendo los valores de entrada y obteniendo los resultados esperados. Para la capa de las composiciones, se ha usado BPELUnit para el desarrollo de las pruebas de cada una de las composiciones. A continuación, se definen los «frameworks» anteriores para el desarrollo de las pruebas y se especifican los pasos realizados.

JUnit

Primero de todo, se va a comentar una breve introducción a JUnit y algunos conocimientos básicos que se han de saber. JUnit [25] es un «framework» que permite la realización de la ejecución de clases Java de manera controlada, permitiendo evaluar si el funcionamiento de cada uno de los métodos se comporta como se espera. Es una herramienta de software libre y fue escrito originalmente por Erich Gamma y Kent Beck. Es un ejemplo de la arquitectura xUnit para marcos de pruebas unitarias. También, se puede utilizar para las pruebas de regresión, cuando una parte del código ha sido modificada y sea necesario comprobar que se sigue cumpliendo con todos los requisitos.

En cada prueba se tendrá que realizar una operación y verificar que el resultado devuelto es el esperado, para ello, JUnit dispone de los siguientes métodos para hacer comprobaciones:

- **assertEquals(esperado,real):** Comprueba que esperado sea igual a real. Lanza un AssertionFailedError si no se produce el resultado esperado.
- **assertTrue(condición):** Comprueba que la condición se cumple. Lanza un AssertionFailedError si no se produce el resultado esperado.
- **assertFalse(condición):** Comprueba que la condición es falsa. Lanza un AssertionFailedError si no se produce el resultado esperado.
- **assertNull(objeto):** Comprueba que objeto sea «null».
- **assertNotNull(objeto):** Comprueba que objeto no sea «null».
- **assertSame(objeto_esperado,objeto_real):** Comprueba que objeto_esperado y objeto_real sean el mismo objeto. Lanza un AssertionFailedError si no se produce el resultado esperado.
- **assertNotSame(objeto_esperado,objeto_real):** Comprueba que objeto_esperado no sea el mismo objeto que objeto_real. Lanza un AssertionFailedError si no se produce el resultado esperado.
- **fail():** Devuelve una alerta informando que el «test» ha fallado.

Todos los métodos anteriores admiten opcionalmente un argumento String con un mensaje explicativo de la comprobación que realiza el «test». De estar presente, este argumento sería el primero de la lista, por ejemplo assertTrue (mensaje, condición).

Además, existen 2 métodos que sirven para inicializar cada una de las pruebas y liberar recursos que hagan falta. Estos métodos son: setUp y tearDown. Existen otros 2 métodos más que se ejecutan una sola vez al inicio de los «test» "setUpClass" y una sola vez al finalizar todos los «test» "tearDownClass".

A continuación, se detallan las anotaciones que incluye:

- **@RunWith:** Se le asigna una clase a la que JUnit invocará en lugar del ejecutor por defecto de JUnit.
- **@Before:** Se indica que el siguiente método se debe ejecutar antes de cada «test» (precede al método setUp). Si tiene que preceder al método setUpClass, la notación será "@BeforeClass".
- **@After:** Indica que el siguiente método se debe ejecutar después de cada «test» (precede al método tearDown). Si tiene que preceder al método tearDownClass, la notación será "@AfterClass".
- **@Test:** Indica a JUnit que se trata de un método de «Test». En versiones anteriores de JUnit los métodos tenían que tener un nombre con la siguiente estructura: "test". Con esta notación colocada delante de los métodos se puede elegir el nombre libremente.

Como para este proyecto ha utilizado un inyector de dependencias como Spring, para poder trabajar con JUnit como mínimo se tendría que levantar el contenedor de dependencias. Spring proporciona esta funcionalidad gracias a 2 clases:

- **AbstractTransactionalJUnit4SpringContextTests:** Da soporte para inyección de dependencias.
- **AbstractTransactionalDataSourceSpringContextTests:** A parte de dar soporte para la inyección crea una transacción por cada «test» y hace un «rollback» al terminar, de esta forma no deja información en la Base de Datos.

En ambos casos se tendrá que añadir la anotación @ContextConfiguration con la ruta hasta el fichero de configuración de Spring. De aquí, será donde lea la configuración de los «bean» a cargar. Para poder cargar algún «bean» sólo se tendrá que crear un atributo que se llame igual que el id del «bean» de Spring y añadirle la anotación @Autowired. Por ejemplo:

```
1 @Autowired
2 private ServicioHipoteca servHipoteca;
```

DbUnit

DbUnit [12] es una extensión de JUnit para realizar «test» unitarios que permiten interactuar con un conjunto de datos. Dirigida a proyectos con bases de datos que, entre otras cosas, pone su base de datos en un estado conocido antes y después de cada «test». Esta es una excelente manera de evitar la gran cantidad de problemas que pueden ocurrir cuando un caso de prueba estropea la base de datos y hace que los «test» posteriores fallen o empeoren el daño. DbUnit se basa en un sistema de importación/exportación de los datos de la base de datos a un fichero XML. DbUnit también, puede ayudar a verificar que los datos de base de datos coinciden con un conjunto esperado de valores.

Principalmente, se añadieron las dependencias de JUnit y DbUnit al pom.xml. El siguiente paso, fue crear manualmente un conjunto de datos XML desde cero creando el archivo dataset porque se necesitaban algunos datos para poder trabajar con las pruebas. Seguidamente, se configuró la conexión con la base de datos,

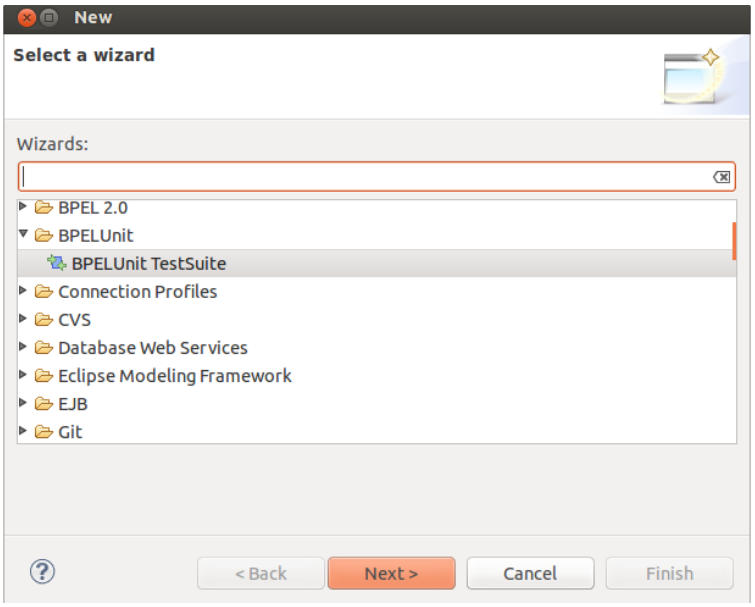


Figura 7.5.: Nuevo fichero BPTS

después, se creó un objeto IDatabaseTester y se inicializó la base de datos antes de la prueba con setDataSet(). A continuación, se usó databaseTester.onSetup() y databaseTester.onTearDown() para inicializar y limpiar, antes y después de cada una prueba. Por último, se implementaron los métodos de prueba como se hace normalmente con JUnit. La base de datos quedaba inicializada antes y limpiada después de cada método de prueba como se ha detallado en pasos anteriores.

BPELUnit

BPEL o WS-BPEL es una lenguaje formal para expresar un proceso de negocio como una composición de servicios web. Para casi todos los lenguajes de programación, existe un marco de pruebas unitarias, por ejemplo JUnit para Java. Estos marcos de pruebas permiten a los desarrolladores probar fácilmente pequeños trozos de código que han escrito. Hasta hace poco, BPEL carecía de ese apoyo. Sin embargo, BPELUnit fue desarrollado para cerrar esta brecha. Véase 1.4.9.

Además de los ficheros WSDL y BPEL correspondientes, se necesita un fichero BPTS que contenga los casos de prueba contra los que ejecutar la composición. Se debe de crear un fichero BPTS para cada composición. Primero, se importan los procesos en Eclipse, creando un nuevo proyecto. Para ello selecciona: File → New → Project. A continuación, en el asistente que aparece, se elige Project en la pestaña General y se le da un nombre al nuevo proyecto, TestsTransaccionesTerravis. Se crea una nueva carpeta para cada proceso y en cada una se importa la carpeta build generada por Intalio.

Se pulsa con el botón derecho del ratón encima de cada una de las carpetas del proyecto y se selecciona: New → Other. De esta manera, aparece de nuevo un asistente donde se elige BPELUnit TestSuite en la pestaña BPELUnit (Figura 7.5). Seguidamente, se le asigna un nombre al nuevo fichero BPTS y se consigue tener en nuestro proyecto un nuevo fichero BPTS ya listo con el que trabajar. Se dispone de una interfaz con la que se puede añadir los distintos casos de pruebas. En la figura 7.6 se puede un ejemplo de uno de los ficheros.

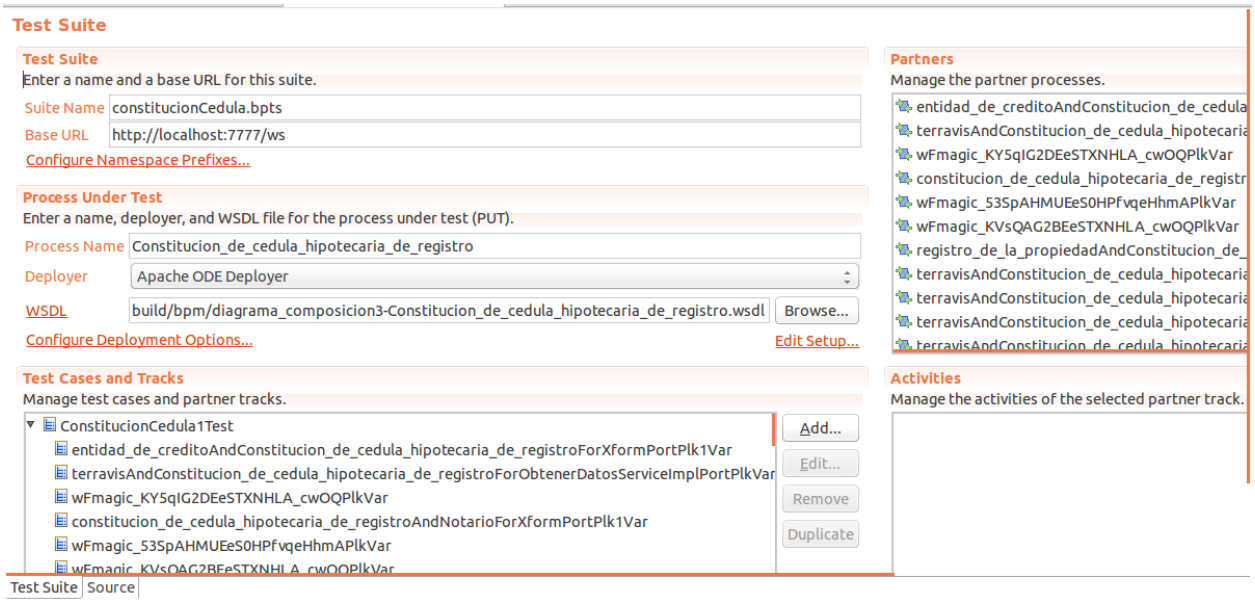


Figura 7.6.: Muestra fichero BPTS

7.6. Ejecución de las pruebas

Una vez se tuvieron creadas todos los «test» se ejecutaron. Se pueden observar en la tabla 7.1 algunos resultados obtenidos, se centra en algunos fallos detectados gracias a las pruebas con JUnit y DbUnit, que fueron solucionados:

Clases	Métodos	Fallo y solución
ServicioFirmanteTest	testSaveFirmante()	No se esperaba la excepción invalid.ereid al introducir un dato de tipo String con caracteres guiones. Se modificó la expresión regular de @Pattern de la clase Hipoteca porque el carácter guión estaba en la expresión pero no lo captaba.
ServicioPlazoTest	testSavePlazo()	No se esperaba una excepción porque había fallos en las funciones “boolean isIBANValido(final String iban)” y “int calculoDigitosIBAN(final String codigoPais, final String idBanco, final String numCuenta)” de la clase Utilidades. Por ejemplo, se corrigieron algunas extracciones de caracteres que no eran correctas.

Clases	Métodos	Fallo y solución
ServicioInmuebleTest	testSaveInmueble()	No se esperaba una excepción porque había fallos en las funciones "boolean isEgridValido(final String egrid)" y "int calculoDigitosEGRID(final String numeroSinDc)" de la clase Utilidades. Por ejemplo, se tuvo que añadir ceros por la izquierda porque sino al calcular los dígitos y ponerlos en el String había uno menos.
ServicioHipotecaTest	testSaveHipoteca()	No se esperaba una excepción porque había fallos en las funciones "boolean isEreidValido(final String ereid)" y "int calculoDigitosEreid(final String ereidSinDc)" de la clase Utilidades. Se corrigieron.
ServicioPersonaFisicaTest	testSavePersonaFisica()	No se esperaba una excepción porque había fallos en las funciones "boolean isNIPValido(final String nip)" y "int calculoDigitoControlNIP(final String nipSinDc)" de la clase Utilidades. Se corrigieron.
ServicioPersonaJuridicaTest	testSavePersonaJuridica()	No se esperaba una excepción porque había fallos en las funciones "boolean validateIDE(final String ide)" y "int calculoDigitoControlIDE(final String ideSinDc)" de la clase Utilidades. Por ejemplo, se tuvo que añadir el «Pattern» en la función en este caso porque no se podía controlar desde los BO.
ServicioEntidadCreditoTest	testSaveEntidadCredito()	No se esperaba una excepción porque había fallos en las funciones "boolean validateIDE(final String ide)" y "int calculoDigitoControlIDE(final String ideSinDc)" de la clase Utilidades. Se corrigieron.
ServicioNotarioTest	testSaveNotario()	No se esperaba una excepción porque había fallos en las funciones "boolean validateIDE(final String ide)" y "int calculoDigitoControlIDE(final String ideSinDc)" de la clase Utilidades. Se corrigieron.

Clases	Métodos	Fallo y solución
ServicioRegistroPropiedadTest	testSaveRegistroPropiedad()	No se esperaba una excepción porque había fallos en las funciones boolean validateIDE(final String ide)" y "int calculoDigitoControlIDE(final String ideSinDc)" de la clase Utilidades. Se corrigieron.

Tabla 7.1.: Tabla de algunos fallos detectados tras las pruebas DbUnit y JUnit.

Se pueden observar en la tabla 7.2 algunos resultados obtenidos, se centra en algunos fallos detectados gracias a las pruebas manuales a los servicios web y que fueron solucionados:

Clases	Métodos	Fallo y solución
DocumentoServiceImpl	retrieveAll, retrieveById, retrieveByRegistroPropiedad, retrieveByPromesaPago	Todos los objetos de la lista aparecía el atributo numDoc a cero. El error es que faltaba inicializar este atributo en todos los objetos. Se solucionó añadiendo el valor al atributo.
EntidadCreditoServiceImpl	retrieveAll	Salía error no controlado si el atributo idPromesaPago era nulo. El error es que faltaba controlar que cuando idPromesaPago sea nulo que no inicialice el atributo del objeto con su id. Se solucionó controlando con una condición, cuando sea nulo que no añada el valor al atributo.
HipotecaServiceImpl	update	No actualizaba. El error es que se había utilizado el método «create» del servicio en vez de «update» y se solucionó modificándolo.
PersonaFisicaServiceImpl	retrieveAll, retrieveById, retrieveByLugar, retrieveByComunidad,	Salía error no controlado si el atributo NIP_comunidad era nulo. El error es que faltaba controlar que cuando NIP_comunidad sea nulo que no inicialice el atributo del objeto con su nip. Se solucionó controlando con una condición, cuando sea nulo que no añada el valor al atributo.
RegistroPropiedadServiceImpl	update	No actualizaba. El error es que no se le introducía a numPart el valor del parámetro de entrada. Se solucionó introduciendo dicho valor.

Clases	Métodos	Fallo y solución
ComprobarDatosServiceImpl	validacionInmueble	Con los dos parámetros vacíos devolvía «true» y tenía que devolver «false». Modificado el código para que lo mostrara controlando los parámetros vacíos.
ComprobarDatosServiceImpl	aprobacionRegistrarHipoteca	Si el parámetro de entrada se dejaba vacío, devolvía «true» y tenía que devolver «false». Modificado el código para que lo mostrara controlando el parámetro vacío.
ComprobarDatosServiceImpl	validarEntidadCreditoAntigua	Mostraba un error no controlado cuando entidad era nula y quería acceder a sus atributos. Modificado el código para controlarlo.
RegistrarDatosServiceImpl	registrarCambioAcreedor	No controlaba bien el antiguo acreedor porque introducía uno de los antiguos acreedores como último antiguo acreedor y entonces había registrado dos nuevos acreedores. Se debía de controlar que el antiguo acreedor sea el actual que financia el último con su fchFin vacía.

Tabla 7.2.: Tabla de algunos fallos detectados tras las pruebas manuales a los servicios web.

Una vez se tuvieron creadas todos los «test» de BPELUnit se ejecutaron. Se pueden observar en la tabla 7.3 algunos resultados obtenidos, se centra en algunos fallos detectados gracias a las pruebas con BPELUnit, que fueron solucionados:

Prueba	Fallo	Solución
ConstitucionCedula1Test	Fallo de conversión	Se ha modificado la variable condición que estaba declarada int y debía ser String.
CambioAcreedor1Test	Fallo de asignación	Asignada correctamente el atributo numPart en UpdateEntidad.
PrevioComunidadTest	Fallo al almacenar los firmantes tanto en proceso 1 y 2	Modificada la expresión XQuery para las referencias, inicializando también los firmantes, ya que al volver a empezar el bucle no se almacenaban los firmantes y se perdían, almacenando siempre sólo dos firmantes.

Prueba	Fallo	Solución
CambioAcreedor1Test	Fallo al registrar los plazos en las dos ramas	No se registraban bien los plazos, tras rechazar la primera promesa se queda con los últimos plazos de la promesa aprobada. Por lo tanto, en el registro se tuvo que cambiar el contador inicial y el contador final para que se realizara correctamente.

Tabla 7.3.: Tabla de algunos fallos detectados tras las pruebas BPELUnit.

7.7. Detalles de las pruebas

En la implementación de las pruebas se han de destacar distintos detalles importantes a tener en cuenta:

- **Correlación:** Se ha añadido correlación a los archivos BPEL para cada proceso porque parece que Intalio lo hace internamente y ha sido necesario para el correcto funcionamiento de las pruebas.
- **Fallo ODE:** Fallo encontrado en las pruebas al volver a reinicializar una variable con un literal y ha habido que añadir código a los archivos BPEL para cada proceso para las pruebas.
- **Condiciones de carrera entre las actividades del fichero de extensión BPTS y BPEL:** Al ejecutar las pruebas y observar el archivo catalina.out, se comprobó que llegaba el mensaje del `<receive>` antes que la ejecución del propio `<receive>`. Es mejor que primero ODE llegue al `<receive>` y luego se le mande el mensaje. Se hacía primero un «invoke» (cuyo receiveSend tiene un id en el fichero de extensión BPTS) y luego, un par receive/reply (cuyo sendReceive tiene un dependsOn en el id anterior en el fichero de extensión BPTS). En la práctica, es mejor introducir una espera de 1 segundo entre el receiveSend y el sendReceive del fichero de extensión BPTS. Para eso, BPELUnit tiene un tipo de actividad llamado «Wait»: se añadió después del receiveSend. Con esto se asegura de que primero entre en el `<receive>` y después, se mande el mensaje.

  Al ser las pruebas muy largas, algunos de los receiveSend del final de las pruebas fallaban. Eso era debido a que todos las partnerTrack se inician a la vez, si no se pone un dependsOn antes, los receiveSend finales de la prueba esperan desde el inicio de la prueba, y si la prueba supera los 25 segundos en total sufren un «timeout». Para corregir esto, a los últimos receiveSend se les ha puesto que dependan de un receiveSend que siempre va a venir antes. Algunas pruebas son demasiado largas y aunque dependan del último receiveSend que viene antes sufren un «timeout» y entonces, se ha tenido que aumentar en BPELUnit esos 25 segundos a 40 segundos.

8. Conclusiones

En este capítulo se expondrán las conclusiones del proyecto. A continuación, se va a ver los objetivos alcanzados, las conclusiones personales con los conocimientos adquiridos y las futuras mejoras.

8.1. Objetivos alcanzados

Actualmente, se puede confirmar que se han cumplido todos los objetivos que se querían conseguir en este proyecto:

1. **La capa de persistencia y de dominio.** Este sistema tiene su propia lógica de negocio y capa de persistencia que encapsula el modo en que los datos se almacenan y se recuperan de una base de datos relacional. Se puede almacenar, actualizar, consultar o borrar toda la información necesaria en la Base de Datos y así, gestionarla fácilmente.

2. **Los servicios web.** Se han desarrollado los servicios necesarios, divididos en 3 modelos de servicios, unos para cada una de las entidades de negocio, otros para cada una de las tareas necesarias en el proceso de negocio y otro que no cubren una necesidad concreta de negocio pero que se dedican a proporcionar funcionalidad de utilidad como gestionar la seguridad de acceso a aplicaciones o servicios de la plataforma.

3. **Las composiciones.** Se han desarrollado las 3 composiciones WS-BPEL. Estas se componen de todos los ficheros necesarios y cumplen todos los requisitos para formar parte del repositorio de composiciones WS-BPEL que gestiona el grupo UCASE de la Universidad de Cádiz.

8.2. Conclusiones personales

La realización de este PFC ha requerido mucha dedicación, aprendizaje y esfuerzo por mi parte. Este proyecto ha sido el más importante de la carrera, ya que, ha tenido mucha complejidad y se ha realizado durante un largo periodo de tiempo. Esta experiencia, me ha servido tanto para investigar y obtener bastantes conocimientos nuevos de distintas tecnologías, lenguajes de programación y herramientas, como para organizar y planificar un proyecto de más envergadura. Se ha dedicado un poco más de tiempo de lo esperado para investigarlas y practicarlas un poco antes de comenzar el desarrollo de este proyecto. También, se han tenido que afianzar conocimientos obtenidos durante los años universitarios, han sido necesarias las siguientes asignaturas: Ingeniería del Software, Bases de Datos, Programación Orientada a Objetos, Estructuras de Datos, etcétera. Se comenzó de cero, sólo se conocían las funcionalidades del sistema a través de la documentación de un caso de estudio llevado a cabo en Suiza, así que, se ha empleado toda la Ingeniería del Software.

En resumen, he obtenido abundantes conocimientos sobre los servicios web, he aprendido nuevos lenguajes de programación, como: Java y WS-BPEL. C/C++ era el único lenguaje de programación que sabía, aunque Java está basado en C++, hizo falta aprendizaje. Con Java se ha desarrollado parte del código fuente del proyecto y la otra, con WS-BPEL, un lenguaje que desconocía totalmente, que no se parecía en nada a los lenguajes que me habían enseñado hasta el momento, complicado de aprender y de usar. Además, ha sido necesario aprender el resto de tecnologías y lenguajes para la realización de las distintas composiciones,

como WSDL, XQuery, XPath, XML Schema, SOAP, BPMN y BPELUnit. He aprendido a trabajar en un entorno de integración continua (Maven) con dependencias entre varios proyectos. Ahora me desenvuelvo más ágilmente con la herramienta Eclipse. Ha sido utilizada para la creación de la capa de persistencia, de dominio y API.

Además, he podido aprender a utilizar diferentes herramientas, como Intalio para el desarrollo de las composiciones, MySQL workbench para el desarrollo de la base de datos, además del sistema de control de versiones (Subversion) para subir las modificaciones del proyecto a la forja del grupo UCASE. También, hay que destacar Sonar y Jenkins porque han sido bastante útil para tener un código de calidad. Referente a las pruebas unitarias, se ha usado DbUnit, JUnit y BPELUnit para mejorar la calidad del producto resultante. Por otro lado, es la primera vez que realizo una memoria para un proyecto como este. He aprendido a utilizar la herramienta Kile que junto con LaTeX se ha desarrollado esta documentación de forma rápida, fácil y estructurada.

Además, decir que se ha realizado una labor de autoaprendizaje y comprensión, ya que, la mayoría de técnicas, lenguajes y distintas tecnologías han sido estudiadas a través de documentación, con manuales, y se ha decidido que el proyecto se va a publicar como código libre bajo la «Apache Software License 2.0» [48] siguiendo con la misma línea de los proyectos del grupo UCASE y ser útil para la comunidad de desarrolladores. Referente al grupo UCASE, es la primera vez que he participado en un grupo de investigación y ha sido muy beneficioso.

Para concluir, me quedo con lo más importante que es haber superado todos los errores que me han ido surgiendo durante el desarrollo de este proyecto para así, aprender de ellos y en un futuro, no volver a cometerlos. Ha sido un camino largo y difícil pero me siento satisfecha del trabajo realizado y del aprendizaje adquirido.

8.3. Trabajo futuro

Como se comentó en el primer capítulo de estas memorias, el caso de estudio se simplificó para adaptarlo a este proyecto. La aplicación Terravis, se compone de muchos procesos más sobre la gestión inmobiliaria. Al realizar este PFC una única persona y en un tiempo determinado, se tuvo que simplificar y se tomó la decisión de realizar 3 procesos completos. Como trabajo futuro se podrían implementar muchos procesos más necesarios para la gestión inmobiliaria. Por lo tanto, se ampliaría la base de datos si hiciera falta y se realizarían más servicios web si las nuevas composiciones lo necesitaran.

También los procesos realizados se han simplificado, la firma de los documentos no se han llevado a cabo mediante la firma electrónica, guardando solamente la fecha de la firma del documento. Por lo tanto, en un futuro se podría adaptar estos 3 procesos llevados a cabo en este proyecto para que pudieran realizarla.

A. Manual de usuario

En este manual se van a describir las distintas funciones que proporciona Intalio|BPMS y se va a explicar detalladamente todos los pasos a seguir con cada composición teniendo en cuenta los perfiles o roles existentes, con el objetivo que los usuarios puedan sacarle el máximo partido.

A.1. Uso para un Usuario

Un usuario tiene el permiso de iniciar un proceso, realizar tareas y recibir notificaciones, sólo podrá acceder si está registrado en el sistema. El usuario abrirá una nueva ventana para colocar el siguiente URL: `http://localhost:8080/ui-fw` e introduce su usuario y su «password» como aparece en la figura A.1. Aparecerá la pantalla «Intalio Workflow User Interface», es la interfaz mediante la cual, los usuarios iniciarán los procesos y llevarán a cabo sus tareas. En la fase de definición del proceso, se diseñarán las tareas y el flujo de información de los usuarios.

Uso dependiendo del rol:

A.1.1. Entidad de crédito

Se introduce el usuario: «examples\bbiedermann» y la contraseña: «password». En la vista de procesos, los procesos posibles para iniciar son:

1. **Registrar cédula hipotecaria.** Está activo en la tabla con la lista de procesos. Seguidamente, continúa realizando una tarea, en la cual, aparece un formulario donde se debe seleccionar la variante de acuerdo para comprobar si la declaración va a ser establecida previamente o directamente y pulsar «start».

2. **Constitución de cédulas hipotecarias de registro.** Está activo en la tabla con la lista de procesos. Seguidamente, continúa realizando una tarea, en la cual, aparece el formulario "orden electrónica constitución". Se tiene que rellenar dicho formulario y pulsar «complete».

Uso del proceso «Registrar cédula hipotecaria»

En la vista de tareas (figura A.2) o notificaciones según el orden, las tareas y notificaciones posibles a realizar son:

1. Si el usuario ha seleccionado previo:
   - Tarea **Elección tipo propietario**. Aparece un formulario que permite seleccionar para que tipo de propietario se desea realizar dicha declaración. Dependiendo de lo que se seleccione, en la siguiente tarea aparecerá un formulario distinto.
   - Si se selecciona Persona Física:

     Tarea **Acuerdo persona física**. Aparece el formulario "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" para las personas físicas (Si fuera una comunidad, se registraría cada una de las personas físicas que la componen). Se tiene que rellenar dicho formulario y pulsar «complete».

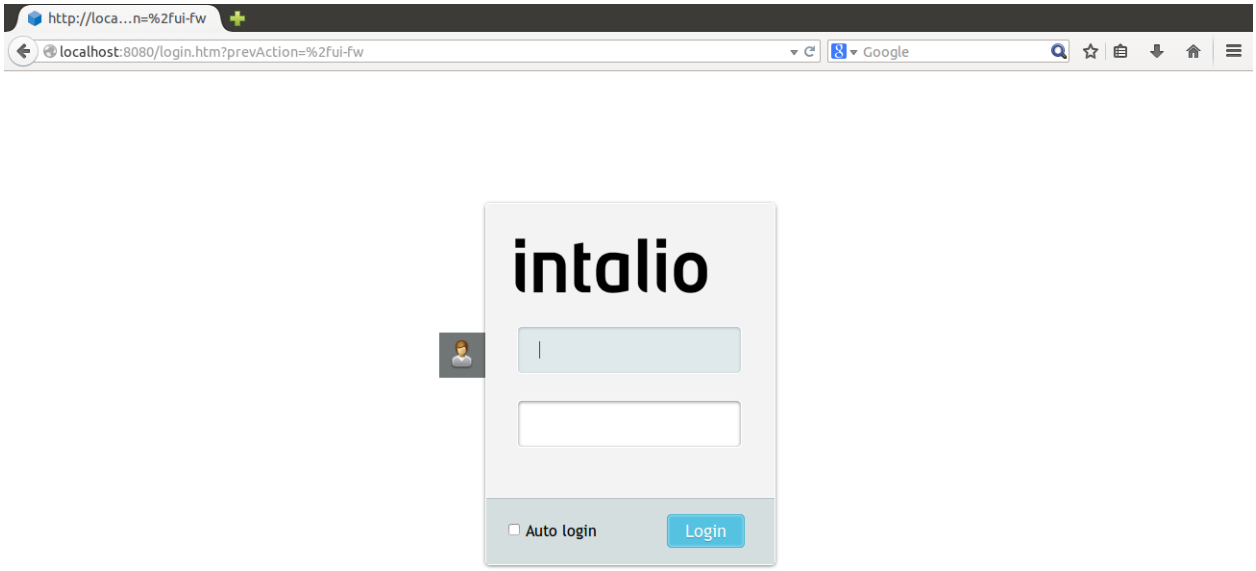


Figura A.1.: Intalio - Login del usuario.

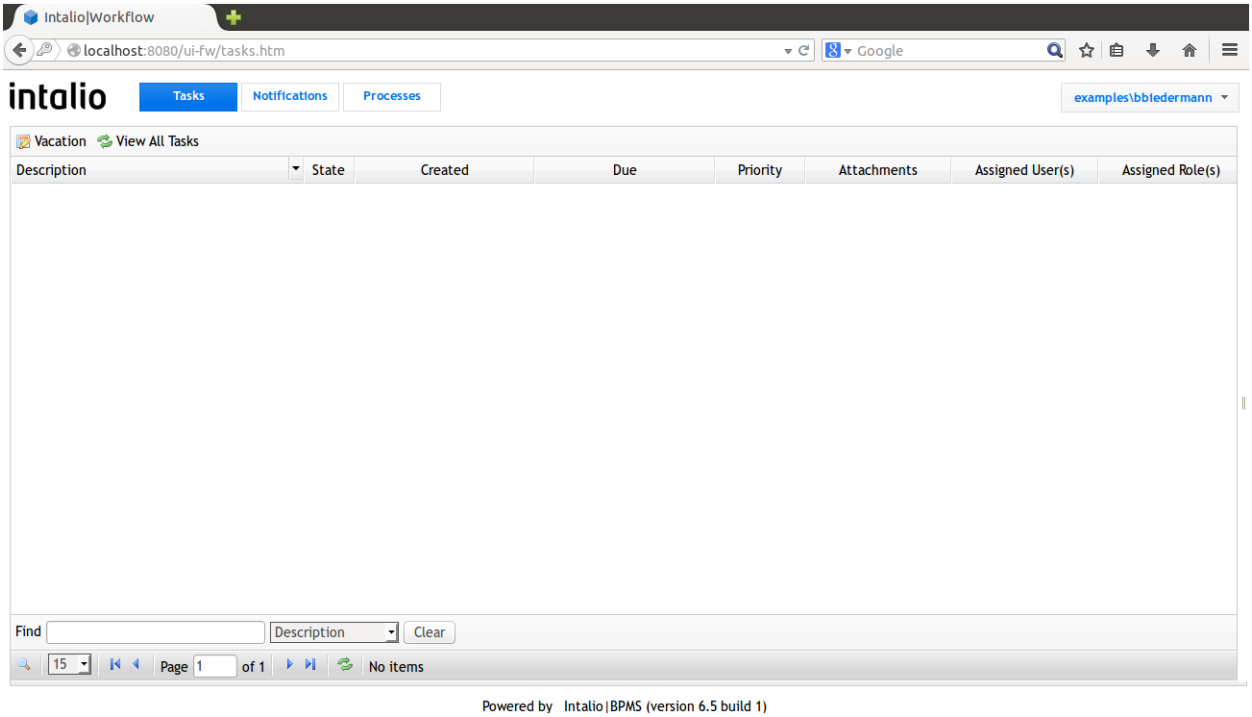


Figura A.2.: Intalio|Workflow - Vista de tareas.

- Si se selecciona Persona Jurídica:

  Tarea **Acuerdo persona jurídica**. Aparece el formulario "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" para personas jurídicas. Se tiene que rellenar dicho formulario y pulsar «complete».

- Tarea **Elección acuerdo**. Aparece un formulario que permite seleccionar el tipo de propietario para el que se desea realizar dicha declaración o continuar con el proceso.

2. Tarea **Validar inmueble**. Muestra un nuevo formulario donde se debe seleccionar lo que se va a introducir, el número del inmueble o EGRID e introducirlo en el campo de al lado. Además, debe seleccionar si quiere seguir introduciendo o continuar con el proceso. Dichos datos son para comprobar si existe el inmueble y poder continuar el proceso. Se rellena el formulario y se pulsa «complete». Esta tarea se puede repetir hasta que se pulse continuar con el proceso o hasta que algún inmueble no sea válido.

3. **Notificación EC**. Si apareciera alguna notificación se le indica mediante un formulario que el inmueble no es válido y que no se puede continuar con el proceso o que no se puede continuar con el proceso Registrar cédula hipotecaria porque se produjo una excepción en la validación del inmueble.

4. Tarea **Orden entidad de crédito**. Aparece el formulario "orden entidad de crédito", se introducen los datos, se selecciona el tipo de propietario y se pulsa «complete».

5. Tarea **Orden inmueble**. Aparece el formulario "orden entidad de crédito (inmuebles)", se introducen los datos y se pulsa «complete».

6. **Notificación EC**. Si no estuvieran en el mismo distrito el sistema finaliza el proceso (envía una notificación y el motivo de la finalización del proceso) o que no se puede continuar con el proceso Registrar cédula hipotecaria porque se produjo una excepción en la validación del mismo distrito.

7. Si el usuario ha seleccionado en la orden como tipo de propietario comunidad,

   Tarea **Orden comunidad**. Muestra el formulario "orden entidad de crédito (comunidad)", se rellenan los datos y pulsar «complete».

8. Si el usuario ha seleccionado en la orden como tipo de propietario persona física,

   Tarea **Orden persona física**. Muestra el formulario "orden entidad de crédito (persona física)", se rellenan los datos y se debe seleccionar si se desea introducir otro tipo de propietario o continuar con el proceso.

9. Si el usuario ha seleccionado en la orden como tipo de propietario persona jurídica,

   Tarea **Orden persona jurídica**. Muestra el formulario "orden entidad de crédito (persona jurídica)", se rellenan los datos y se debe seleccionar si se desea introducir otro tipo de propietario o continuar con el proceso.

10. Tarea **Orden referencia**. aparece el formulario "orden entidad de crédito (referencia)", se introducirán los datos y pulsar «complete».

11. Tarea **Notificación EC**. Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Registrar cédula hipotecaria porque se produjo una excepción al obtener el número del usuario.

12. Si el usuario ha seleccionado directo al iniciar el proceso,

- Tarea **Elección tipo propietario**. Aparece un formulario que permite seleccionar para que tipo de propietario se desea realizar dicha declaración. Dependiendo de lo que se seleccione, en la siguiente tarea aparecerá un formulario distinto.
- Si se selecciona Persona Física,

  Tarea **Acuerdo persona física**. Aparece el formulario "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" para las personas físicas (Si fuera una comunidad, se registraría cada una de las personas físicas que la componen). Se tiene que rellenar dicho formulario y pulsar «complete».
- Si se selecciona Persona Jurídica,

  Tarea **Acuerdo persona jurídica**. Aparece el formulario "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" para personas jurídicas. Se tiene que rellenar dicho formulario y pulsar «complete».
- Tarea **Elección acuerdo**. Aparece un formulario que permite seleccionar el tipo de propietario para el que se desea realizar dicha declaración o continuar con el proceso.

13. Tarea **Notificación EC**. Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Registrar cédula hipotecaria porque se produjo una excepción al obtener el número del participante y de la transacción.

14. Tarea **Envío de notificación**. Consiste en introducir los datos que faltan en el formulario "Envío de notificación sobre la transformación de una hipoteca".

15. **Notificación EC**. Si ya estuviera registrada la hipoteca, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Registrar cédula hipotecaria porque se produjo una excepción al registrar los datos.

16. **Notificación confirmación registro EC**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

Uso del proceso «Cambio de acreedor»

En este proceso participan dos tipos de entidades de crédito, según cada una de ellas, en la vista de tareas o notificaciones según el orden, las tareas y notificaciones posibles a realizar son:

- Si se selecciona registrar la cédula hipotecaria,
  1. Tarea **Aprobar o no aprobar dicha promesa de pago**. Aparece el formulario "decisión promesa de pago" y se puede aprobar o no. Si se aprueba, se continúa con el proceso, si no, se repetirán de nuevo desde la tarea que contiene el formulario "promesa de pago" hasta que se apruebe.
  2. **Notificación ECant**. Se lleva a cabo la comprobación interna de validar la fecha de los plazos, si no fuera correcta aparecerá en la vista de notificaciones una notificación con el motivo del rechazo. Si se produce una excepción aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al validar la fecha de los plazos.
  3. **Notificación ECant**. Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al obtener los números de la transacción, promesa y participantes.

4. **Notificación ECant**. Si no es aprobado el registro de la hipoteca, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al registrar los datos.
5. **Notificación confirmación registro ECant**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

- Si se selecciona cédula hipotecaria ya está registrada,
  1. Tarea **Orden del cambio de acreedor para la entidad de crédito**. Aparece el formulario "cambio de acreedor", se introducen los datos y se pulsa «complete».
  2. **Notificación ECant**. Se lleva a cabo la comprobación interna de validar la entidad de crédito vendedora, si no fuera correcta aparecerá en la vista de notificaciones una notificación con el motivo del rechazo. Si las entidades de crédito no tuvieran el mismo registro de la propiedad, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción en la validación de la entidad de crédito vendedora y del mismo registro de la propiedad.
  3. Tarea **Aprobar o no aprobar dicha promesa de pago**. Aparece el formulario "decisión promesa de pago" y se puede aprobar o no. Si se aprueba, se continúa con el proceso, si no, se repetirán de nuevo desde la tarea que contiene el formulario "promesa de pago" hasta que se apruebe.
  4. **Notificación ECant**. Se lleva a cabo la comprobación interna de validar la fecha de los plazos, si no fuera correcta aparecerá en la vista de notificaciones una notificación con el motivo del rechazo. Si se produce una excepción aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al validar la fecha de los plazos.
  5. **Notificación ECant**. Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al obtener los números de la transacción, promesa y participantes.
  6. **Notificación ECant**. Si no es aprobado el cambio de acreedor, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al registrar los datos.
  7. **Notificación confirmación registro del cambio de acreedor ECant**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

Uso del proceso «Constitución de cédulas hipotecarias de registro»

En la vista de tareas o notificaciones según el orden, las tareas y notificaciones posibles a realizar son:

1. Tarea **Notificación EC**. Si se produce una excepción aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Constitución de cédula hipotecaria de registro porque se produjo una excepción al obtener el código de activación.

2. Tarea **Notificación EC**. Si se produce una excepción aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Constitución de cédula hipotecaria de registro porque se produjo una excepción al obtener los números de los participantes y de la transacción.

3. **Notificación EC**. Si no estuviera registrada la hipoteca, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Constitución de cédula hipotecaria de registro porque se produjo una excepción al registrar los datos.

4. **Notificación confirmación registro EC**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

A.1.2. Entidad de crédito compradora

Se introduce el usuario: «examples\mboss» y la contraseña: «password». En la vista de procesos, los procesos posibles para iniciar son:

1. **Cambio de acreedor.** Está activo en la tabla con la lista de procesos (Siempre lo inicia la entidad de crédito compradora o nueva). Seguidamente, continúa realizando una tarea, en la cual, aparece un formulario donde se debe introducir la variante inicial. Se pueden seleccionar dos opciones de cambio de acreedor, una para registrar la cédula hipotecaria y en la que el registro de la propiedad pertenece al sistema, y otra donde la cédula hipotecaria ya está registrada y en la que el registro de la propiedad pertenece al sistema. Pulsar «start».

Uso del proceso «Cambio de acreedor»

En este proceso participan dos tipos de entidades de crédito, según cada una de ellas, en la vista de tareas o notificaciones según el orden, las tareas y notificaciones posibles a realizar son:

- Si se selecciona registrar la cédula hipotecaria,
    1. Tarea **Variante acuerdo**. Se le aparece un formulario donde se debe introducir la variante del acuerdo para comprobar si la declaración del acuerdo va a ser establecida previamente o directamente y pulsar «start».
    2. Si el usuario ha seleccionado previo,
        - Tarea **Elección tipo propietario**. Aparece un formulario que permite seleccionar para que tipo de propietario se desea realizar dicha declaración. Dependiendo de lo que se seleccione, en la siguiente tarea aparecerá un formulario distinto.
        - Si se selecciona Persona Física:

            Tarea **Acuerdo persona física**. Aparece el formulario "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" para las personas físicas (Si fuera una comunidad, se registraría cada una de las personas físicas que la componen). Se tiene que rellenar dicho formulario y pulsar «complete».
        - Si se selecciona Persona Jurídica:

            Tarea **Acuerdo persona jurídica**. Aparece el formulario "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" para personas jurídicas. Se tiene que rellenar dicho formulario y pulsar «complete».

- Tarea **Elección acuerdo**. Aparece un formulario que permite seleccionar el tipo de propietario para el que se desea realizar dicha declaración o continuar con el proceso.

3. Tarea **Validar inmueble**. Muestra un nuevo formulario donde se debe seleccionar lo que se va a introducir, el número del inmueble o EGRID e introducirlo en el campo de al lado. Además, debe seleccionar si quiere seguir introduciendo o continuar con el proceso. Dichos datos son para comprobar si existe el inmueble y poder continuar el proceso. Se rellena el formulario y se pulsa «complete». Esta tarea se puede repetir hasta que se pulse continuar con el proceso o hasta que algún inmueble no sea válido.
4. **Notificación ECNue**. Si apareciera alguna notificación, se le indica mediante un formulario que el inmueble no es válido y que no se puede continuar con el proceso o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción en la validación del inmueble.
5. Tarea **Orden entidad de crédito**. Aparece el formulario "orden entidad de crédito", se introducen los datos, se selecciona el tipo de propietario y se pulsa «complete».
6. Tarea **Orden inmueble**. Aparece el formulario "orden entidad de crédito (inmuebles)", se introducen los datos y se pulsa «complete».
7. **Notificación ECNue**. Si no estuvieran en el mismo distrito, el sistema finaliza el proceso (envía una notificación y el motivo de la finalización del proceso) o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción en la validación del mismo distrito.
8. **Notificación ECNue**. Si las entidades de crédito no tuvieran el mismo registro de la propiedad, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso) o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción en la validación del mismo registro de la propiedad.
9. Si el usuario ha seleccionado en la orden como tipo de propietario comunidad,

   Tarea **Orden comunidad**. Muestra el formulario "orden entidad de crédito (comunidad)", se rellenan los datos y pulsar «complete».
10. Si el usuario ha seleccionado en la orden como tipo de propietario persona física,

    Tarea **Orden persona física**. Muestra el formulario "orden entidad de crédito (persona física)", se rellenan los datos y se debe seleccionar si se desea introducir otro tipo de propietario o continuar con el proceso.
11. Si el usuario ha seleccionado en la orden como tipo de propietario persona jurídica,

    Tarea **Orden persona jurídica**. Muestra el formulario "orden entidad de crédito (persona jurídica)", se rellenan los datos y se debe seleccionar si se desea introducir otro tipo de propietario o continuar con el proceso.
12. Tarea **Orden referencia**. aparece el formulario "orden entidad de crédito (referencia)", se introducirán los datos y pulsar «complete».
13. Tarea **Notificación ECNue**. Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al obtener el número del usuario.
14. Tarea **Promesa de pago**. Aparecerá una nueva tarea que contiene el formulario "promesa de pago" y se introducen los datos. Tras pulsar nuevamente en «complete».
15. Tarea **Plazo**. Va a aparecer otra tarea con el formulario "promesa de pago (plazo)", donde se introducen los plazos de dicha promesa de pago hasta que se seleccione continuar con el proceso y se haga clic en «complete».

16. **Notificación promesa de pago aprobada.**. Para comprobar que se ha aprobado la promesa de pago correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.
17. **Notificación ECNue**. Si la diferencia entre fecha y fecha es mayor que 18 meses, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al validar la fecha de los plazos.
18. **Notificación ECNue**. Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al obtener los números de la transacción, promesa y participantes.
19. Si el usuario ha seleccionado directo al iniciar el proceso,
    - Tarea **Elección tipo propietario**. Aparece un formulario que permite seleccionar para que tipo de propietario se desea realizar dicha declaración. Dependiendo de lo que se seleccione, en la siguiente tarea aparecerá un formulario distinto.
    - Si se selecciona Persona Física,

      Tarea **Acuerdo persona física**. Aparece el formulario "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" para las personas físicas (Si fuera una comunidad, se registraría cada una de las personas físicas que la componen). Se tiene que rellenar dicho formulario y pulsar «complete».
    - Si se selecciona Persona Jurídica,

      Tarea **Acuerdo persona jurídica**. Aparece el formulario "Declaración transformación de una cédula hipotecaria sobre papel en cédula hipotecaria de registro" para personas jurídicas. Se tiene que rellenar dicho formulario y pulsar «complete».
    - Tarea **Elección acuerdo**. Aparece un formulario que permite seleccionar el tipo de propietario para el que se desea realizar dicha declaración o continuar con el proceso.
20. Tarea **Envío de notificación**. Consiste en introducir los datos que faltan en el formulario "Envío de notificación sobre la transformación de una hipoteca".
21. **Notificación ECNue**. Si ya estuviera registrada la hipoteca, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al registrar los datos.
22. **Notificación confirmación registro ECNue**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

- Si se selecciona cédula hipotecaria ya está registrada,
  1. Tarea **Promesa de pago**. Aparecerá una nueva tarea que contiene el formulario "promesa de pago" y se introducen los datos. Tras pulsar nuevamente en «complete».
  2. Tarea **Plazo**. Va a aparecer otra tarea con el formulario "promesa de pago (plazo)", donde se introducen los plazos de dicha promesa de pago hasta que se seleccione continuar con el proceso y se haga clic en «complete».

3. **Notificación ECNue**. Se lleva a cabo la comprobación interna de validar la fecha de los plazos, si no fuera correcta aparecerá en la vista de notificaciones una notificación con el motivo del rechazo. Si se produce una excepción aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al validar la fecha de los plazos.
4. **Notificación ECNue**. Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al obtener los números de la transacción, promesa y participantes.
5. **Notificación ECNue**. No debe aparecer ninguna. Si no es aprobado el cambio de acreedor, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al registrar los datos.
6. **Notificación confirmación registro del cambio de acreedor ECNue**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

A.1.3. Registro de la propiedad

Uso del proceso «Registrar cédula hipotecaria»

Se introduce el usuario: «examples\tblum» y la contraseña: «password». En la vista de tareas o notificaciones según el orden, las tareas y notificaciones posibles a realizar son:

1. Tarea **Orden registro de la propiedad**. Se muestra el formulario “orden registro de la propiedad”, se rellenan los datos que faltan y se pulsa «complete».
2. **Notificación RP**. No debe aparecer ninguna. Si ya estuviera registrada la hipoteca, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Registrar cédula hipotecaria porque se produjo una excepción al registrar los datos.
3. **Notificación confirmación registro RP**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

Uso del proceso «Cambio de acreedor»

- Si se selecciona registrar la cédula hipotecaria,
  1. Tarea **Orden registro de la propiedad**. Se muestra el formulario “orden registro de la propiedad”, se rellenan los datos que faltan y se pulsa «complete».
  2. **Notificación RP**. Si no es aprobado el registro de la hipoteca, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al registrar los datos.

3. **Notificación confirmación registro RP**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

- Si se selecciona cédula hipotecaria ya está registrada,
  1. Tarea **Orden cambio de acreedor**. Se muestra el formulario "confirmación registro", se rellenan los datos que faltan y se pulsa «complete».
  2. **Notificación RP**. Si no es aprobado el cambio de acreedor, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Cambio de acreedor porque se produjo una excepción al registrar los datos.
  3. **Notificación confirmación registro del cambio de acreedor RP**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

Uso del proceso «Constitución de cédulas hipotecarias de registro»

En la vista de tareas o notificaciones según el orden, las tareas y notificaciones posibles a realizar son:

1. Tarea **Confirmación inscripción**. Se muestra el formulario "confirmación electrónica inscripción", se rellenan los datos que faltan y se pulsa «complete».

2. **Notificación RP**. Si no estuviera registrada la hipoteca, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Constitución de cédula hipotecaria de registro porque se produjo una excepción al registrar los datos.

3. **Notificación confirmación registro RP**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

A.1.4. Notario

Uso del proceso «Constitución de cédulas hipotecarias de registro»

Se introduce el usuario: «examples\smorgan» y la contraseña: «password». En la vista de tareas, las tareas posibles a realizar son:

1. Tarea **Requisa constitución**. Se muestra el formulario "Requisa electrónica creación de una cédula hipotecaria de registro", introduce el código de activación y adjunta PDF que es el contrato de garantía escaneado. Por último, hace clic en «complete».

2. Tarea **Notificación N**. Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Constitución de cédula hipotecaria de registro porque se produjo una excepción al obtener los números de los participantes y de la transacción.

3. Tarea **Documento**. Se muestra el formulario para introducir los datos del contrato de garantía y adjuntar un PDF que es dicho contrato de garantía escaneado. Se puede seleccionar para seguir introduciendo más documentos o para continuar con el proceso. Por último, hace clic en «complete».

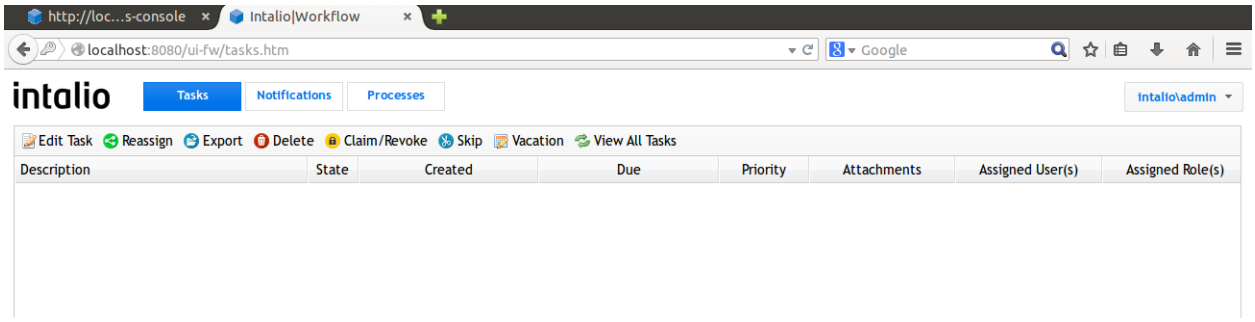


Figura A.3.: Intalio|Workflow - Adinistrador.

4. **Notificación N**. Si no estuviera registrada la hipoteca, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si ya estuviera registrada la transacción, el sistema termina la conexión (envía una notificación y el motivo de la finalización del proceso). Si se produce una excepción, aparece una notificación con el fallo concreto o que no se puede continuar con el proceso Constitución de cédula hipotecaria de registro porque se produjo una excepción al registrar los datos.

5. **Notificación confirmación registro N**. Para comprobar que todo se ha realizado correctamente, debe aparecer una notificación de confirmación. Se pulsa «dimiss» si se quiere eliminar dicha notificación.

A.2. Uso para el Administrador del sistema

El usuario que posea un perfil de administrador del sistema tendrá todos los permisos existentes. Primero, el desarrollador debe de haber desplegado el proceso. Si se accede a `http://localhost:8080/ui-fw` con el administrador puedo ver todos los procesos, tareas y notificaciones de todos los roles. Se introduce el usuario por defecto: «admin» y la contraseña: «changeit». Se puede ver en la figura A.3 la pantalla «Intalio Workflow User Interface», donde el administrador iniciará los procesos, llevará a cabo tareas e incluso podrá reasignarlas dichas tareas a otros usuarios o roles y podrá ver las notificaciones.

Si se accede al administrador web del servidor de la URL: `http://localhost:8080/bpms-console`, se introduce el usuario por defecto: «admin» y la contraseña: «changeit». En la instalación de «Intalio BPMS Server» se encuentra «Intalio BPMS Console». «Intalio BPMS Console» permite acceder y administrar los procesos desplegados, además de lanzar los procesos en el servidor. También se puede, iniciar, activar, desactivar y suspender o terminar procesos en ejecución.

En la consola se tienen tres vistas diferentes: Procesos («Processes»), Instancias («Instances») y Herramientas («Tools»).

Vista de procesos

En la figura A.4 se muestra una lista de los procesos que han sido desplegados en el servidor. Para cada uno de ellos se puede ver:

- **«Lifecycle»:** Indica si el proceso está activo (listo para instanciar) o retirado (desplegado pero no se puede instanciar).
- **«In Progress»:** Número de instancias del proceso ejecutándose.
- **«Failure»:** Número de instancias del proceso en estado fallido.
- **«Suspended»:** Número de instancias suspendidas.

- **«Failed»:** Número de instancias del proceso que han fallado.
- **«Terminated»:** Número de instancias del proceso que han terminado.
- **«Completed»:** Número de instancias del proceso que se han completado.
- **Total:** Número total de instancias.

En la vista de procesos se tienen otros botones que permiten:

- **«Start»:** Envía una instancia del proceso.
- **«Activate»:** Cambia su ciclo de vida de retirado a activo.
- **«Retire»:** Cambia su ciclo de vida de activo a retirado.
- **«Desploy»:** Despliega un proceso.
- **«Undeploy»:** Elimina un proceso desplegado.

Si desde esta vista se accede al detalle de un proceso, se obtienen 3 vistas:

- **«Info»:** Muestra la información del proceso.
- **«Diagram»:** Muestra el diagrama del proceso de negocio.
- **«Resources»:** Muestra los recursos utilizados por el proceso.

Vista de instancias

Muestra el estado de las instancias de los procesos, un histórico de la actividad del proceso y además, permite iniciar instancias, suspender, reanudar la ejecución de instancias suspendidas y eliminarlas. Se ofrece una búsqueda avanzada para definir filtros que permiten obtener vistas personalizadas, además de poder guardar estos filtros como búsquedas predefinidas.

Si desde esta vista se accede a una instancia de proceso determinado, se mostrará la definición, a los datos, a sus variables, y a un histórico detallado de los eventos ocurridos durante la ejecución del proceso. La vista de instancias se puede ver en la figura A.5.

Vista de herramientas

En la figura A.6 se puede ver que en la vista de herramientas se tiene acceso a la configuración de «logging». Se muestra una interfaz en la que se puede configurar el nivel de «loggin» para cada clase java desplegada en el «server».

A.3. Organización y asignación de tareas a recursos

Intalio permite definir cualquier número de usuarios y asignarle roles. Un usuario puede tener asignados cualquier número de roles. Se realiza mediante un fichero XML, llamado security.xml, que se encuentra en intalio-bpms-6.5.1/var/config.

En este fichero pueden definirse tres tipos de elementos: «realm», «user» y «role». «Realm» permite agrupar usuarios y roles dentro de un mismo ámbito, sobre el que puede trabajar un proceso, y permite crear un espacio de nombres con el que identificar a los diferentes recursos. Para cada usuario definido dentro de un «realm» se define un identificador que servirá como nombre de usuario para el «login», además del nombre, «email» y «password» del mismo. También, se define aquí el listado de roles asignados al

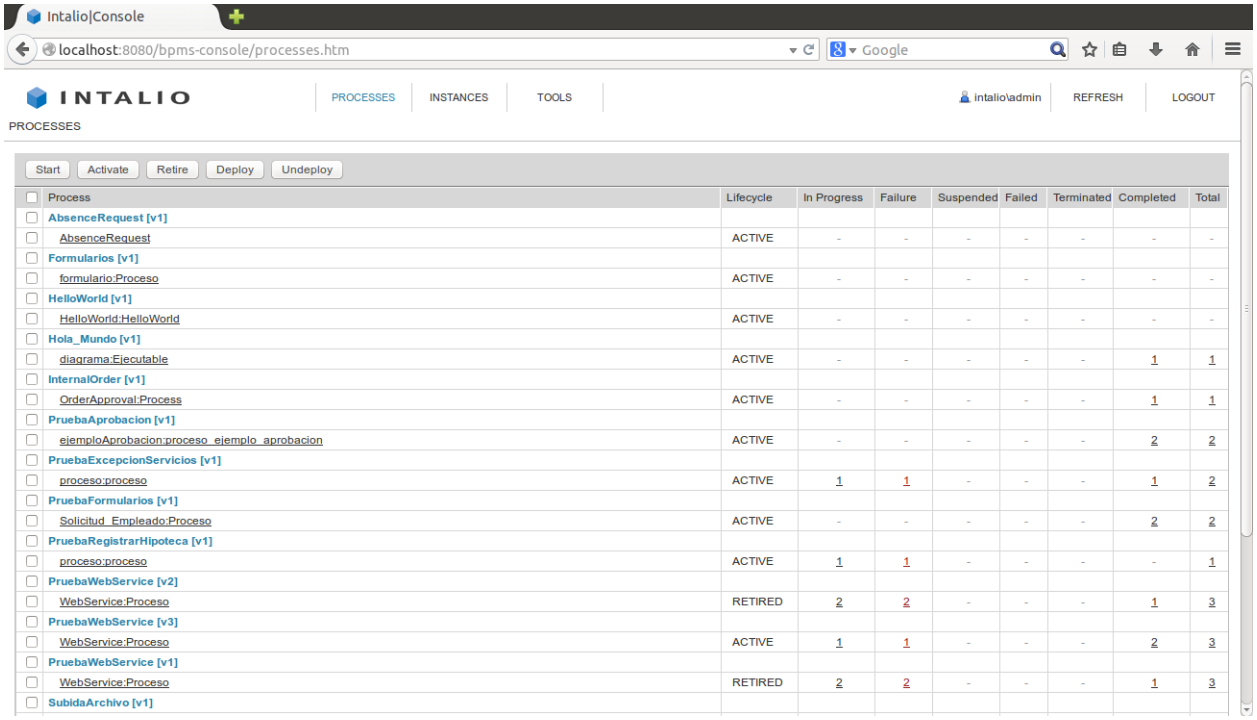


Figura A.4.: Intalio|Console - Vista de procesos.

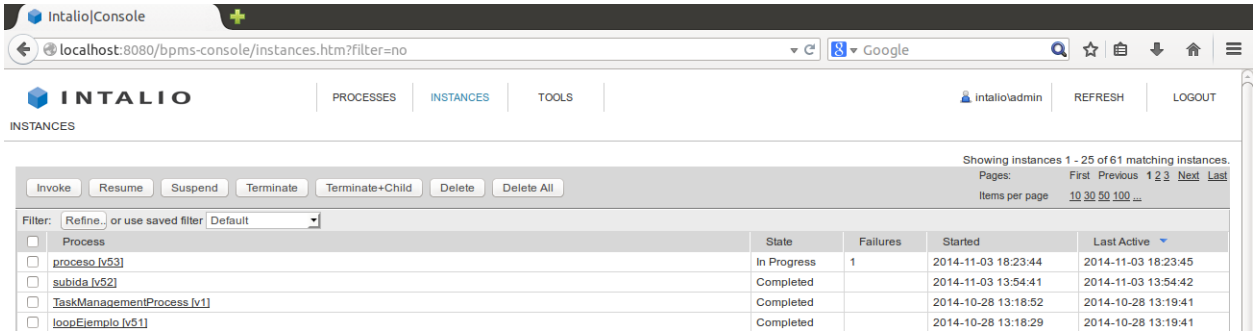


Figura A.5.: Intalio|Console - Vista de instancias.

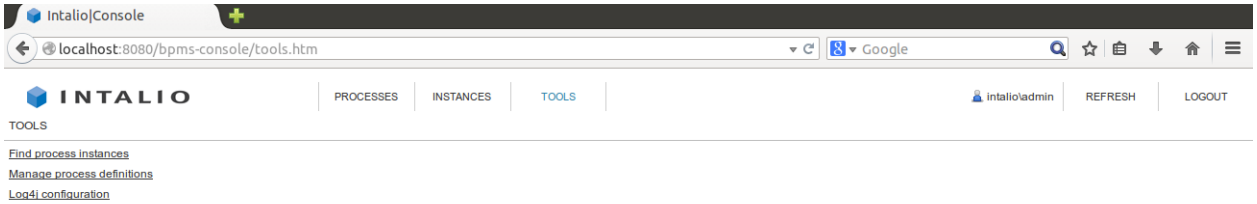


Figura A.6.: Intalio|Console - Vista de herramientas.

usuario, ya sea correspondiente al mismo «realm» o a otro. La definición de cada rol incluye igualmente un identificador y una descripción, y puede definir roles heredados, dando lugar a una jerarquía de roles.

Para este proyecto se han creado los roles para entidad de crédito, entidad de crédito compradora, notario y registro de la propiedad, cuyos identificadores son entcredito, entcreditocomp, notario y regpropiedad. También, se han creado los usuarios Birgit Biedermann cuyo identificador es bbiedermann y rol es entcredito, Martin Boss cuyo identificador es mboss y rol es entcreditocomp, Susan Morgan cuyo identificador es smorgan y rol es notario y Theodor Blum cuyo identificador es tblum y rol es regpropiedad.

Finalmente, se ha modificado el administrador, asignándole estos nuevos roles para poder reasignar las tareas en ejecución. Las tareas pueden asignarse a cualquier rol, mostrándose en ese caso a todos los usuarios que tengan dicho rol asignado. También, es posible asignar a un usuario concreto, referenciado directamente por su nombre.

B. Manual del desarrollador

En este capítulo se van a indicar los pasos a seguir para poder instalar en nuestro equipo todo el software necesario indicado a lo largo de esta memoria. Estas instrucciones han sido elaboradas para un sistema Ubuntu 13.10. En cuanto al hardware en el que se ha desarrollado y ejecutado el proyecto tiene memoria de 5,7 GiB, procesador: Intel R Core TM i5-3337U CPU @ 1.80GHz x 4, gráficos: Intel R Ivybridge Mobile, tipo de Sistema Operativo: 64-bit y disco: 345,3 GB. Este anexo está orientado a desarrolladores que quieran realizar alguna modificación al código.

B.1. Instalación de herramientas

Antes de la descarga del código fuente se va a explicar como se instalan las herramientas que se necesitan usar para poder realizar cambios al código.

B.1.1. Maven

Introducir en la terminal:

```
$ sudo apt-get install maven
```

Comprobar la versión que se ha instalado:

```
$ mvn -version
Apache Maven 3.0.4
Maven home: /usr/share/maven
Java version: 1.7.0_51, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-7-openjdk-amd64/jre
Default locale: es_ES, platform encoding: UTF-8
OS name: "linux", version: "3.11.0-12-generic", arch: "amd64", family: "unix"
```

B.1.2. JDK de Java

Introducir en la terminal:

```
$ sudo apt-get install openjdk-6-jdk
```

Comprobar la versión que se ha instalado:

```
$ java -version
java version "1.6.0_31"
OpenJDK Runtime Environment (IcedTea6 1.13.3) (6b31-1.13.3-1ubuntu1~0.13.10.1)
OpenJDK 64-Bit Server VM (build 23.25-b01, mixed mode)
```

B.1.3. Subversion

Introducir en la terminal:

```
$ sudo apt-get install subversion
```

Comprobar la versión que se ha instalado:

```
$ svn --version
svn, version 1.7.9 (r1462340) compilado Oct 15 2013, 12:40:34

Copyright (C) 2013 The Apache Software Foundation.
This software consists of contributions made by many people; see the NOTICE
file for more information.
Subversion is open source software, see http://subversion.apache.org/

...
```

B.1.4. Eclipse JEE

Eclipse no se puede instalar desde la terminal, se tiene que descargar de su página oficial, en descargas:

`http://www.eclipse.org/downloads/`

Se baja la última versión disponible de «Eclipse Integrated Development Environment (IDE) for Java EE Developers», se extrae el archivo comprimido descargado en la ubicación que se desee y se puede ejecutar el programa haciendo doble clic en el ejecutable de Eclipse. Finalmente, aclarar que para la realización de este proyecto, la versión concreta que se ha usado es Kepler.

B.1.5. TomEE +

Para empezar, se tiene que descargar de su página oficial, en descargas:

`http://tomee.apache.org/downloads.html`

Se baja la última versión disponible. Una vez descargado, se extrae el archivo comprimido en la ubicación que se desee.

Es momento de integrar Eclipse JEE con TomEE. Se realiza de una forma muy sencilla, en la pestaña de «servers», pulsando botón derecho «New» → «Server». Al pulsar esta opción, se elige el tipo de servidor que se desea añadir al entorno de desarrollo, en este caso se selecciona Apache Tomcat 7 y se puede cambiar el nombre del servidor si lo desea. Elegido el tipo de servidor, se solicita que se especifique en qué directorio se encuentra, en este caso, donde se ha descomprimido el paquete TomEE descargado. Se selecciona el directorio y se pulsa "«Finish»". En la pestaña de servidores aparecerá el nuevo servidor para que se pueda utilizar.

B.1.6. Intalio|Designer

Para revisar el diseño del proceso de caso de estudio, se requiere tener instalado el diseñador de proceso de intalio Intalio-Designer-6.5.1. Descargar Intalio BPMS de:

`http://www.intalio.com/resources/downloads/`

"Intalio|BPMS Community Edition". Primero, se tiene que registrar en la web, se hace clic en «register», se introduce el «E-mail», se pulsa «register», de nuevo, se rellenan los datos y se introduce una contraseña. Una vez concluido el registro, se recibe un correo para corroborar dicho «E-mail». Finalmente, se hace clic en "«download»", se inicia la sesión con el «E-mail» y la contraseña y se observa lo que aparece en la figura B.1.

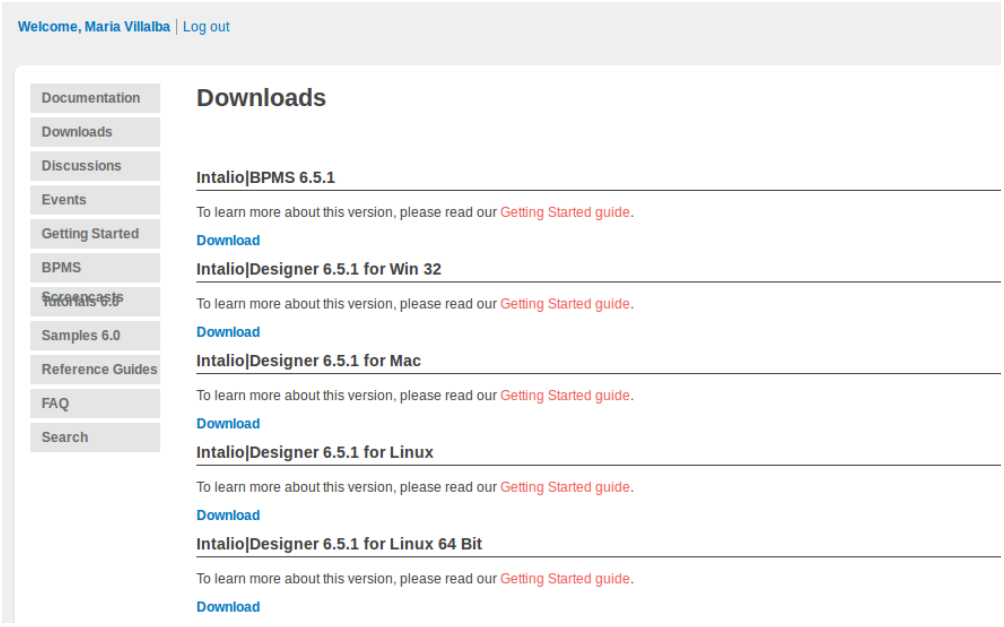


Figura B.1.: Descargas Intalio.

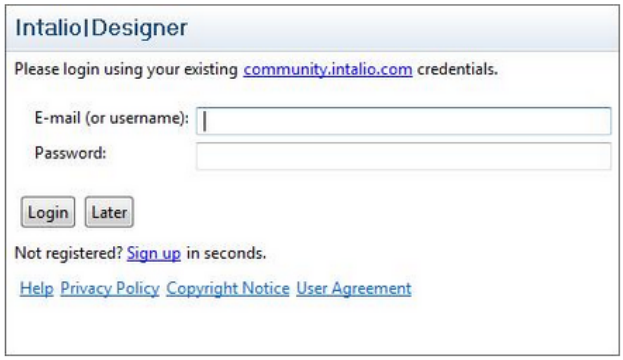


Figura B.2.: Identificación en la comunidad de Intalio.

Se pulsa "«Download»" tanto en Intalio|BPMS 6.5.1, que es el servidor, y en «Intalio|Designer 6.5.1 for Linux 64 bit» ya que es para 64 bit. En este apartado se habla de la instalación de Intalio|Designer. Se descomprime el archivo anteriormente descargado "com.intalio.bpms.designer.distribs.ce.linux_64-6.5.1.tgz" y aparece la carpeta "designer.ce.linux-6.5.1". Se abre y ejecuta "«designer»".

Primero de todo, se tiene que crear un «workspace», similar a como se hace con Eclipse para empezar a utilizar esta herramienta. De esta forma se habilita una zona de trabajo en la que guardar los elementos que se generan. Se pedirá identificación para saber si se pertenece a la comunidad de Intalio mediante el «E-mail» y un «password» como se observa en la figura B.2. Llegados a este punto, se puede dar de alta en la comunidad, introducir nuestras credenciales o bien decir que ya se hará, por lo que comenzará a cargar la aplicación. Una vez hecho esto se puede empezar a utilizar la herramienta.

Conexión MySQL con Intalio-bpms

1. **Paso 1.** Descarga del «JDBC Driver for MySQL». Para poder utilizar desde Intalio-bpms el gestor de base de datos MySQL a través de JDBC, es necesario configurar lo que se conoce como el Conector J para MySQL, es decir, una librería proporcionada por Oracle para el manejo de las funciones de acceso a datos con dicho motor. En la página `http://www.mysql.com/products/connector/`

se puede encontrar la página de descarga. Si se pulsa en "«Download»" se ve una nueva pantalla desde donde se puede seleccionar dos archivos: "mysql-connector-java-5.1.25.tar.gz" y "mysql-connector-java-5.1.25.zip". Se selecciona el primero de ellos, desde donde finalmente se obtiene el archivo comprimido. se descomprime el archivo, con lo cual se añade una carpeta denominada "mysql-connector-java-5.1.25". Dentro de esta carpeta, entre otros archivos, existe uno denominado "mysql-connector-java-5.1.25-bin.jar", este archivo contiene la librería que tendrá que conocer el IDE de Intalio-bpms para poder utilizar el acceso a MySQL.

2. **Paso 2.** Se inicia el servicio de la BD que se va a utilizar, se revisan los datos en la BD para saber que se va a hacer y sobre qué. Los procesos son modelados desde intalio|Designer, pero ejecutados desde intalio|Server, por lo tanto, para que este pueda realizar una conexión satisfactoria, se debe incluir en la carpeta "intalio-bpms-6.5.1/common/lib" el driver de conexión a la BD que se está utilizando. Por lo tanto, se copia el archivo "mysql-connector-java-5.1.25-bin.jar" que se encuentra dentro de la carpeta "mysql-connector-java-5.1.25" y se pega en la carpeta "intalio-bpms-6.5.1/common/lib". Ahora, se puede realizar la conexión desde intalio|Designer y realizar el proceso.

3. **Paso 3.** Desde Intalio|Designer se debe crear un nuevo conector a base de datos. En la perspectiva "«Database Development»", se pulsa "«New Connection Profile»".

4. **Paso 4.** Se selecciona la BD deseada y se le pone un nombre. En este caso MySQL.

5. **Paso 5.** Se pulsa "«New Driver Definition»", se selecciona el «driver» de conexión (que se acaba de mover dentro de la carpeta "intalio-bpms-6.5.1/common/lib". Primero, se selecciona: "MySQL JDBC Driver | MySQL | 5.1" y se le pone un nombre si se quiere en "driver name".

6. **Paso 6.** Se pulsa "«Jar List»". Se elimina el «driver» por defecto "remove JAR/ZIP" y se agrega "add JAR/ZIP" el localizado en "intalio-bpms-6.5.1/common/lib". Se hace clic en ok.

7. **Paso 7.** Se debe especificar la base de datos que se va a utilizar en la ventana que se ve en la figura B.3. En "«Database»:" se escribe el nombre de nuestra BD.

8. **Paso 8.** Se corrige el final de la URL quitando «Database» y poniendo nuestro nombre introducido. Finalmente, se pone el «password» del «root» (marcar «save password») para permitir a intalio|designer hacer modificaciones y accesos sobre la base de datos.

9. **Paso 9.** Se hace clic en «Test Connection» para probar que se ha definido correctamente la conexión. Debe aparecer una ventana que diga que es correcto, se le da a ok. Finalmente, se pulsa en "«Finish»". Ya se tiene la base de datos conectada como se puede ver en la figura B.4.

A la hora de crear un conector en Intalio para poder conectar con la base de datos MySQL, se tenía instalado Intalio designer para un sistema operativo de 64 bits para que correspondiera con las características del ordenador pero aparecía un error Standard Widget Toolkit (SWT), se intentó solucionar pero había problemas de compatibilidad porque contenía un xulrunner para 32 bits. Se decidió instalar la versión para un sistema operativo de 32 bits ejecutándolo desde un sistema de 64 bits. Detalles a destacar en la instalación:

- intalio designer de 32 bits (había que poner java 6 (sudo apt-get install oracle-java6-installer, se retoca haciendo gedit designer ini y se ha puesto la ruta de xulrunner)
- Maquina virtual de java de 32 bits
- Bibliotecas necesarias de 32 bit para xulrunner
- Registrado xulrunner con: LD_LIBRARY_PATH=. sudo ./xulrunner-bin –register-global

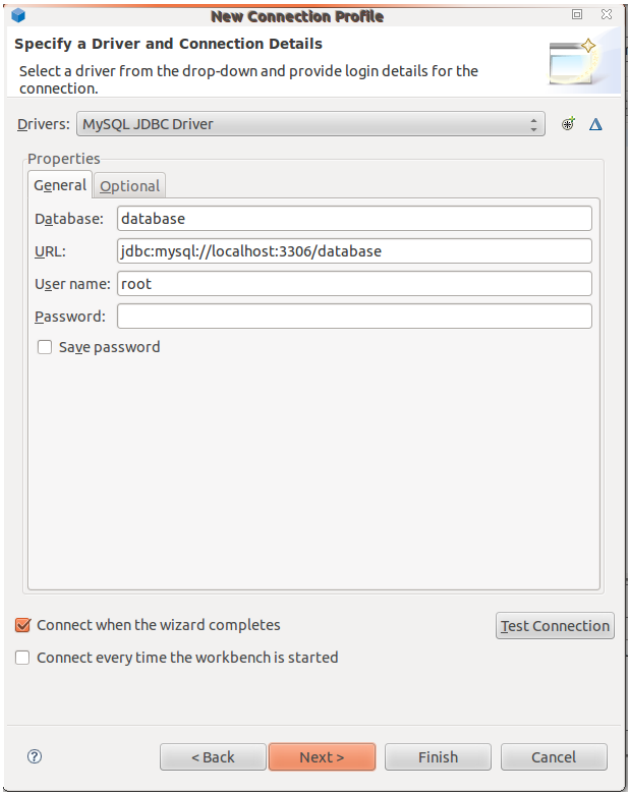


Figura B.3.: Nuevo conector a base de datos.

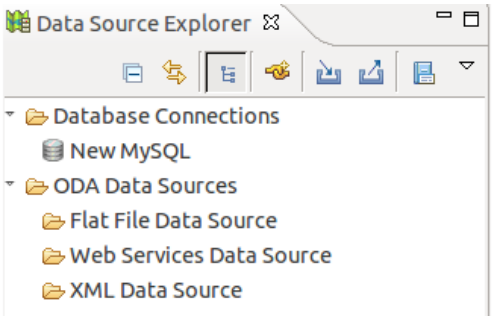


Figura B.4.: Perspectiva Database Development en Intalio|Designer.

B.1.7. Servidor Intalio

Lo primero de todo, se debe abrir una consola y descomprimir el archivo anteriormente descargado "intalio-bpms-6.5.1.zip". Aparece la carpeta "intalio-bpms-6.5.1" e ir al directorio bin.

```
$ cd bin
$ ./startup.sh
bash: ./startup.sh: Permiso denegado
```

Se debe hacer clic en el botón izquierdo en "startup.sh", pulsar permisos y poner "permitir ejecutar el archivo como un programa".

```
$ ./startup.sh
Cannot find ./catalina.sh
This file is needed to run this program
```

Se debe hacer clic en el botón izquierdo en "catalina.sh", pulsar permisos y poner "permitir ejecutar el archivo como un programa".

```
$ export CATALINA_HOME=~/Escritorio/PFC/Programas/intalio-bpms-6.5.1
$ ./startup.sh
Using CATALINA_BASE:   /home/mery/Escritorio/PFC/Programas/intalio-bpms-6.5.1
Using CATALINA_HOME:   /home/mery/Escritorio/PFC/Programas/intalio-bpms-6.5.1
Using CATALINA_TMPDIR: /home/mery/Escritorio/PFC/Programas/intalio-bpms-6.5.1
/temp
Using JRE_HOME:        /usr/lib/jvm/java-7-openjdk-amd64
Using CLASSPATH:       :/home/mery/Escritorio/PFC/Programas/intalio-bpms-
6.5.1/bin/jcl104-over-slf4j-1.4.3.jar:/home/mery/Escritorio/PFC/Programas/
intalio-bpms-6.5.1/bin/slf4j-api-1.4.3.jar:/home/mery/Escritorio/PFC/
Programas/intalio-bpms-6.5.1/bin/slf4j-log4j12-1.4.3.jar:/home/mery/
Escritorio/PFC/Programas/intalio-bpms-6.5.1/bin/log4j-1.2.15.jar:/home/
mery/Escritorio/PFC/Programas/intalio-bpms-6.5.1/var/log:/home/mery/
Escritorio/PFC/Programas/intalio-bpms-6.5.1/bin/bootstrap.jar
```

Se accede al administrador web del servidor con la URL: `http://localhost:8080/bpms-console`. Se introduce el usuario por defecto: «admin» y la contraseña: «changeit». Cuando se entre con los usuarios poner la URL: `http://localhost:8080/ui-fw` (los usuarios no tienen permiso para iniciar sesión en la consola de BPMS que está en `http://localhost:8080/bpms-console`, lo que requiere el papel «ProcessManager»). Los usuarios por defecto son los siguientes:

«Username»: "examples\msmith"; «Password»: "password"
«Username»: "examples\ewilliams"; «Password»: "password"

«role» examples\manager para examples\ewilliams
«role» examples\employee para examples\msmith

Si se entra en `http://localhost:8080/ui-fw` con el administrador se puede hacer todos los procesos tanto del «manager», del «employee», etc.

B.1.8. MySQL

Introducir en la terminal:

```
$ sudo apt-get install mysql-client mysql-server
```

Seguidamente, se introduce la clave de ubuntu. Más adelante, se aprieta la tecla s y «enter». Se escribe la contraseña que se quiere para el usuario principal («root»). Se pulsa la tecla «enter», se reescribe la contraseña y se vuelve a pulsar «enter».

Para finalizar, se instala una herramienta gráfica de MySQL "«workbench»". Introducir en la terminal:

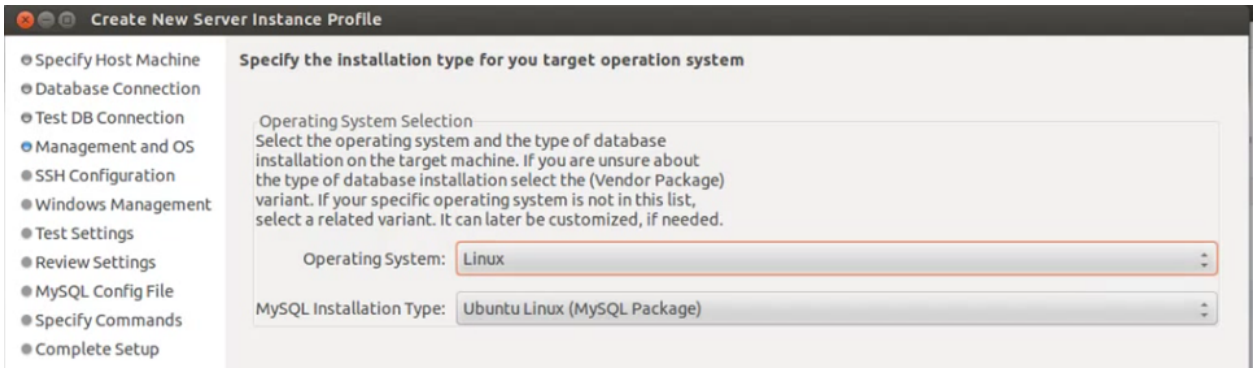


Figura B.5.: Configuración MySQL Workbench.

```
$ sudo apt-get install mysql-workbench
```

Se pulsa la tecla s y «enter». Para conectarse a «mysql server» introducir lo siguiente en la terminal seguido de la contraseña:

```
$ mysql -u root -p
```

Configuración MySQL Workbench

Para configurar MySQL Workbench se pulsa en "«New Server Instance»". Se crea uno nuevo, aparece una ventana donde se selecciona "«localhost»", se le da a «Next». Seguidamente, aparecen una serie de datos rellenos donde solamente se tiene que escribir la contraseña del «root» de la base de datos anteriormente instalada. Se pulsa "«Store in keychain...»", se pone el «password» y se pulsa ok. Se pulsa «Next» y se reescribe de nuevo el «password» del «root». De nuevo se pulsa «next» y se deben poner los datos que se ven en la figura B.5. Se hace clic en «Next» y debe aparecer en pantalla "«Testing host machine settings is done.»". Se le da a «Next», en la nueva ventana que aparece se pulsa "«continue»" y finalmente, se hace clic en "«Finish»". Finalmente, ya se ha agregado el administrador del servidor de MySQL. Se debe crear un nuevo esquema llamado TransaccionesTerravis y al iniciar el servidor desde Eclipse se crean las clases automáticamente.

B.1.9. BPELUnit

BPELUnit se puede instalar desde Eclipse. Para los usuarios, la forma más sencilla de instalar BPELUnit es utilizar el sitio de actualizaciones. Se selecciona «Help» → «Install New Software...», aparece una ventana y se hace clic en "«Available Software Sites»". Se hace clic en "«Add...»" y se introduce `http://update.bpelunit.net` como la ubicación del nuevo sitio. Se hace clic en Aceptar. Una vez que se añade la actualización del sitio BPELUnit, simplemente se marca BPELUnit en «Available Software», se hace clic en Instalar y se siguen las instrucciones. Después de que Eclipse se haya reiniciado, BPELUnit debe estar instalado.

B.2. Descarga y preparación del proyecto

Primero, se descarga el código fuente del proyecto, que se encuentra en el repositorio, con Subversion introduciendo en la terminal lo siguiente:

```
$ svn checkout https://neptuno.uca.es/svn/pfc-mvillaba/trunk/
```

Se obtienen todos los ficheros de la última versión del proyecto, está formado por:

- persistence: Contiene el código de la capa de persistencia y de dominio.
- ws-2-api: Carpeta vacía, se pensó poner los archivos wsdl generados pero finalmente se cogió el URL directamente. Se mantiene por si se quisieran guardar.
- ws-2-impl: Contiene el código de la capa API.
- intalio-projects: Contiene el código de la capa de composición.
- TestsTransaccionesTerravis: Contiene los archivos de extensión BPTS y el código de las composiciones modificado y adaptado para la ejecución de las pruebas.

Para explorar el código de la capa de persistencia y de dominio dirigirse a la carpeta src. El código está organizado de la siguiente manera:

- main/java: Directorio principal con el código Java.
- test/java: Directorio con el código de las pruebas.
- main/resources: Contiene los recursos utilizados por el código principal.
- test/resources: Contiene los recursos necesario para ejecutar las pruebas.

Para explorar el código de la capa API dirigirse a la carpeta src. El código está organizado de la siguiente manera:

- main/java: Directorio principal con el código Java. Donde se encuentran los servicios web divididos en tres carpetas (entidad, tarea, utilidad).
- main/webapp: Contiene los recursos utilizados por el código principal.

Para explorar el código de la capa de composición dirigirse a la carpeta TransaccionesTerravis. El código está organizado de la siguiente manera:

- bpm: Contiene el modelado en BPMN realizado mediante diagramas.
- build: Contiene el código completo generado por Intalio, donde se encuentra el BPEL.
- Esquemas: Contiene los XML schemas creados necesarios en los diagramas.
- Formularios: Contiene los formularios creados necesarios en los diagramas.
- Servicios Web: Contiene los servicios web necesarios en los diagramas.

Antes de realizar los siguientes pasos se explican algunos comandos de maven que pueden servir:

- mvn install: Para instalar el paquete en el repositorio local.
- mvn compile: Para compilar el código fuente del proyecto.
- mvn eclipse:eclipse: Para generar proyectos Eclipse.

- mvn test: Ejecuta todas y cada una de las pruebas unitarias del proyecto.
- mvn package: Genera el artefacto del proyecto.
- mvn deploy: Sube el artefacto a un repositorio Maven en la red.
- mvn clean: Elimina las clases compiladas y los archivos binarios generados del proyecto.
- mvn verify: Verifica el artefacto generado.
- mvn initialize: Configura propiedades y crea directorios.
- mvn validate: Valida el proyecto.

El siguiente paso es descargar todas las dependencias del proyecto utilizando Maven. Desde la terminal, en la carpeta donde se encuentra el proyecto, se introduce lo siguiente:

```
$ mvn compile
```

Después de esto, se crea el proyecto para Eclipse escribiendo en la terminal:

```
$ mvn eclipse:eclipse
```

Se debe importar el proyecto en nuestro Eclipse, se pulsa en «File» → «Import...». Aparece una ventana, se pulsa "General", seguido de "Existing Projects into Workspace" y finalmente, se selecciona «Next». Se deja marcada la opción "Select root directory", se selecciona la ubicación del proyecto y se pulsa "Finish". Puede ser que las variables del «classpath» no estén correctamente definidas y lo indique con una exclamación roja en nuestro proyecto. Para ello. Se pulsa «Window» → «Preferences». Aparece una ventana, se pulsa «Java» → «Build Path» → «Classpath Variables» y se pulsa en "New". Aparecerá un nuevo diálogo; en el campo "Name" se introduce "M2_REPO"; y en el campo "Path", se selecciona el fichero /.m2/repository.

Finalmente, se debe importar intalio-projects en Intalio|Designer, se pulsa en «File» → «Import...». Aparece una ventana, se pulsa "General", seguido de "Existing Projects into Workspace" y finalmente, se selecciona «Next». Seleccione el botón "Select root directory", se selecciona la ubicación del proyecto y se pulsa "Finish".

B.3. Ejecución del proyecto

Para ejecutar nuestro proyecto y procesos creados, es necesario primero configurar para desplegar los procesos, presionando en Intalio|Designer, el botón del engranaje o utilizando el Menú de opciones → «Project» → «Deployment» → «Configure». A continuación, para ejecutar el proyecto bastará hacer clic en el botón «deploy». Es importante verificar que ha sido un éxito su ejecución Se tiene que verificar que nuestro proceso esta en la lista para ser iniciado desde la consola de Intalio|Server. Para iniciar los procesos, antes se debe iniciar el servidor tanto en Intalio como en Eclipse para poder registrar los datos correctamente en nuestra base de datos. Para ejecutar los procesos en Intalio|Server, deben registrarse previamente los datos en la BD de las entidades de crédito, registros de la propiedad, notarios, comunidades, personas físicas, jurídicas y los usuarios que vayan a participar en el proceso de forma manual a través de los servicios web. Se han facilitado algunos datos de ejemplo mediante un archivo que puede ser importado pulsando «Data Import» desde MySQL Workbench.

B.4. Ejecución de las pruebas

Para que los «test» cumplan su función, es muy importante ejecutarlos con mucha frecuencia, especialmente mientras se modifica código.

B.4.1. JUnit

Dentro de la estructura de directorios que poseen los proyectos de maven ya existe una ruta específica para los «test» (“src/test/java”). Una vez se ha descargado el proyecto y compilado con mvn compile, se pueden ejecutar las pruebas escribiendo en la terminal:

```
$ mvn test
```

Desde Eclipse se pueden ejecutar una por una las pruebas manualmente, en la parte izquierda del editor se tiene una sección de exploración de los proyectos abiertos, se despliega el proyecto y la sección src/test/java. Finalmente, hacer clic con el botón derecho en el «test», después en «Run As» y hacer clic en «JUnit Test».

B.4.2. BPELUnit

Para ejecutar el Test Suite, primero se debe asegurar de que el proceso se despliega, es accesible y su BPELUnit Test Suite se guarda. Finalmente, hacer clic con el botón derecho en el fichero con extensión BPTS, se elige «Run as» → «BPELUnit Test Suite». Debe aparecer la vista BPELUnit, en el que se puede observar el progreso de la prueba. Al igual que con la vista JUnit, una barra verde significa que todo ha ido bien y una barra roja significa que una prueba ha fallado.

C. Herramientas

A lo largo de este tercer capítulo, se detallan las distintas herramientas que han sido necesarias utilizar para la realización de este PFC.

C.1. Software utilizado para la gestión de base de datos

C.1.1. MySQL

MySQL [26] es un sistema de gestión de bases de datos relacional, fue creada por la empresa sueca MySQL AB, la cual tiene el «copyright» del código fuente del servidor SQL, así como también de la marca. El lenguaje de programación que utiliza MySQL es SQL que fue desarrollado por International Business Machines (IBM) en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales.

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Para agregar, acceder, y procesar los datos almacenados en una base de datos, se necesita un sistema de administración de bases de datos, tal como MySQL. También, es un sistema de administración relacional de bases de datos. Una base de datos relacional almacena los datos en tablas separadas en lugar de poner todos los datos en un solo lugar. Esto agrega velocidad y flexibilidad. Las tablas son enlazadas al definir relaciones que hacen posible combinar datos de varias tablas cuando se necesitan consultar datos.

MySQL está disponible para múltiples plataformas, sin embargo, las diferencias con cualquier otra plataforma son prácticamente nulas, ya que la herramienta utilizada en este caso es el cliente mysql-client, que permite interactuar con un servidor MySQL (local o remoto) en modo texto. De este modo es posible realizar todos los ejercicios sobre un servidor instalado localmente o, a través de Internet, sobre un servidor remoto.

Se ha elegido este sistema de gestión de base de datos porque MySQL es «Open Source» ya que se puede descargar, usar de forma gratuita y modificarlo, usa la licencia GNU General Public License (GPL) para definir qué es lo que se puede y no se puede hacer con el software para diferentes situaciones, es veloz al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento. Su bajo costo en requerimientos para la elaboración de bases de datos debido a su bajo consumo hace que pueda ser ejecutado en una máquina con escasos recursos sin ningún problema. Se puede configurar e instalar fácilmente, soporta gran variedad de Sistemas Operativos y puede trabajar en distintas plataformas. Tiene baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está. Finalmente, otra de las ventajas son su conectividad, velocidad, y seguridad que hacen de MySQL Server altamente apropiado para acceder bases de datos en Internet.

C.1.2. MySQL workbench

MySQL Workbench [26] es una herramienta visual unificada para los arquitectos de bases de datos, desarrolladores y administradores de bases. MySQL Workbench ofrece modelado de datos, desarrollo de SQL y herramientas completas de administración para la configuración del servidor, la administración de usuarios, copia de seguridad, y mucho más. El espacio de trabajo esta dividido en tres partes:

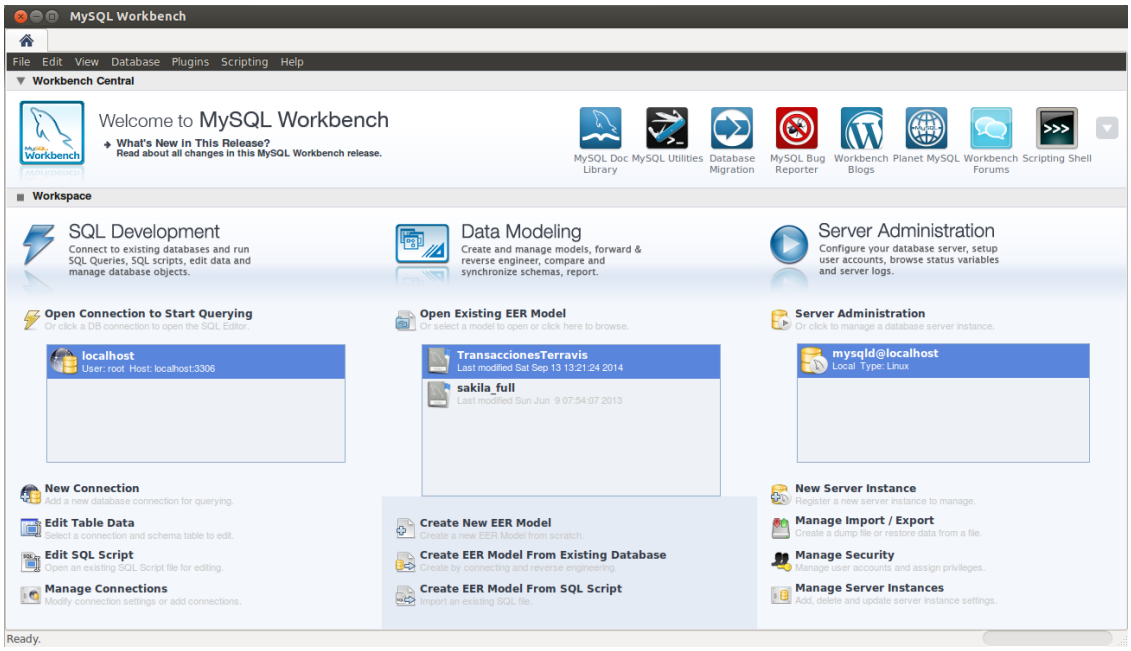


Figura C.1.: Interfaz de MySQL workbench.

1. **Modelado de datos:** Permite al administrados de la base de datos, al desarrollador o arquitecto de datos diseñar, modelar, generar y gestionar bases de datos. Incluye todo lo que un modelador de datos necesita para la creación de modelos ER complejos, ingeniería directa e inversa, y también ofrece funciones clave para la realización de las tareas difíciles de cambio de administrador y de documentación que normalmente requieren mucho tiempo y esfuerzo.

2. **Desarrollo sql:** Proporciona herramientas visuales para crear, ejecutar, y optimizar consultas SQL. El Editor SQL proporciona un color resaltado de sintaxis, auto-completado, la reutilización de fragmentos de código SQL, y el historial de ejecución de SQL. El panel de conexiones de base de datos permite a los desarrolladores gestionar fácilmente las conexiones de base de datos. El buscador de objetos proporciona acceso rápido al esquema y objetos de la base de datos.

3. **Administración del servidor:** Proporciona una consola visual para administrar fácilmente entornos MySQL y obtener una mejor visibilidad en bases de datos. Los desarrolladores y DataBase Administrators (DBAs) pueden utilizar las herramientas visuales para la configuración de los servidores, la administración de usuarios, la realización de copias de seguridad y recuperación, la inspección de los datos de auditoría, y la visualización del estado de base de datos.

En la figura C.1 se observa la interfaz de MySQL workbench.

Esta herramienta se ha utilizado para generar el diagrama de clases una vez creada la base de datos, ver las diferentes tablas creadas en la capa de persistencia y poder comprobar que los datos se insertaban o eliminaban correctamente.

C.2. Software utilizado para la gestión de código

C.2.1. Eclipse JEE

Eclipse IDE for Java EE Developers [20] es una herramienta para desarrolladores Java que permite crear aplicaciones JEE y web, incluyendo un entorno integrado de desarrollo para Java, así como herramientas

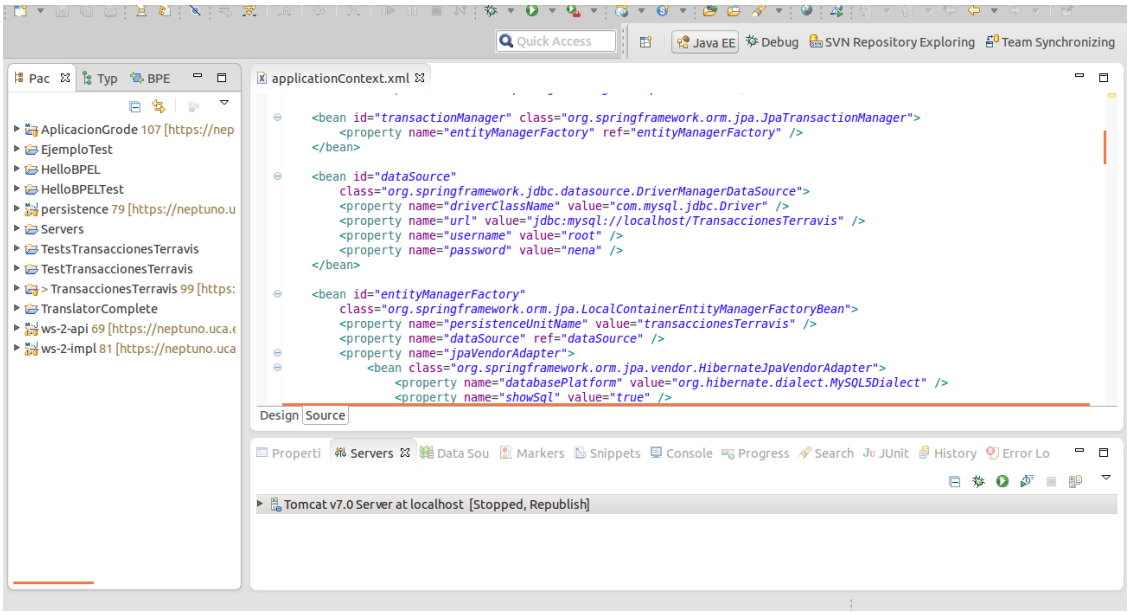


Figura C.2.: Interfaz de Eclipse JEE.

para desarrollo de JEE, JPA, JavaServer Faces (JSF), Mylyn, EGit y otros. Eclipse es un entorno de desarrollo integrado, de código abierto y multiplataforma. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios. En sí mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados («plug-in»). Hay «plug-ins» para el desarrollo de Java (Java Development Tools (JDT)) así como para el desarrollo en C/C++, COBOL, etc. Actualmente, es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (IDE), como el IDE de Java llamado JDT y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes («wizards») para creación de proyectos, clases, «tests», etc., y refactorización. A través de «plugins» libremente disponibles es posible añadir control de versiones con Subversion e integración con Hibernate. Es una potente y completa plataforma de Programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. No es más que un entorno de desarrollo integrado (IDE) en el que encontrarás todas las herramientas y funciones necesarias para tu trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar.

Desde Eclipse IDE for Java EE Developers el programador podrá ir diseñando y trabajando con cada módulo de forma independiente. Para desarrollar cada uno de los aspectos del diseño dispone de un completísimo «pack» de funciones, utilidades y herramientas que le facilitarán todo el trabajo. En la figura C.2 se observa la interfaz de Eclipse JEE.

Esta herramienta se ha utilizado para escribir parte del código fuente del proyecto.

C.2.2. TomEE +

TomEE [49] es un servidor de aplicaciones certificado como Java EE 6 Web Profile compatible y totalmente basado en Tomcat. Primero Apache TomEE es capaz de ejecutar todas las funcionalidades actuales

de Tomcat, tanto JNDI y seguridad. Además es ligero ocupando unos 37MB sin necesitar requerimientos adicionales a los que ya usa Tomcat. Los usuarios de Tomcat estarán totalmente familiarizados.

TomEE (o TomEE Web Profile) contiene:

- **Context and dependency injection (CDI):** Apache OpenWebBeans
- **Enterprise JavaBeans (EJB):** Apache OpenEJB
- **JPA:** Apache OpenJPA
- **JSF:** Apache MyFaces
- **JavaServer Pages (JSP):** Apache Tomcat
- **JSP Standard Tag Library (JSTL):** Apache Tomcat
- **Java Transaction API (JTA):** Apache Geronimo Transaction
- **Servlet:** Apache Tomcat
- **Javamail:** Apache Geronimo JavaMail
- **Bean Validation** - Apache BVal

La distribución TomEE+ (o TomEE Plus) añade lo siguiente:

- **Java API for RESTful Web Services (JAX-RS)** - Apache CXF
- **JAX-WS** - Apache CXF
- **JMS** - Apache ActiveMQ
- **Connector** - Apache Geronimo Connector

Se ha utilizado como servidor en Eclipse para poder probar el proyecto maven creado con la capa de persistencia, de dominio y API.

C.2.3. Intalio|BPMS

Es un software «Open Source» basado en Java-Java 2 Enterprise Edition (J2EE), que implementa BPMS, y está basado en un conjunto de «frameworks» y arquitecturas muy conocidas en la industria del software. Intalio utiliza la notación para diseñar procesos de negocio establecida por el BPMN que puede adaptarse a los requisitos de SOA. Diseñado alrededor de la fuente abierta Eclipse BPMN Modeler, el motor BPEL Apache ODE, y el servicio de tareas WS-Human Tempo desarrollado por Intalio, puede apoyar cualquier proceso, pequeños o grandes.

Intalio|BPMS proporciona todos los componentes necesarios para diseñar, implementar y gestionar los procesos de negocio más complejos, incluida la vigilancia de la actividad empresarial, la gestión de reglas de negocio, gestión de documentos y contenidos, integración de sistemas, protocolos de negocio a negocio y herramientas del portal web.

Características

Intalio|BPMS [24] ofrece una buena experiencia para los usuarios, analistas de procesos, desarrolladores de procesos y administradores. Intalio sigue evolucionando Intalio|BPMS con nuevas funcionalidades y mejoras importantes. Está disponible en dos ediciones: «Community Edition» y «Enterprise Edition».

1. **Modelado de Procesos de Negocio:** Se hace uso de el lenguaje de modelado BPMN, se puede documentar sus modelos de procesos y se pueden exportar y compartir los modelos como imagen o PDF.

2. **Diseño de Procesos de Negocio:** Se diseña formularios Web 2.0 con un editor donde se puede arrastrar y soltar los componentes que se quiera añadir, Formulario Web para móviles conversión automática formulario, se puede definir un flujo de trabajo, se puede definir transformaciones de datos de gran alcance con un editor gráfico: Haga clic y arrastre para el mapeo de datos más rápido, se puede definir relaciones y dependencias con facilidad. Se pueden reutilizar los procesos, los errores se gestionan más rápido y más fácilmente, tiene compensación («roll back») y un editor visual de los artefactos técnicos (WSDL, XML, XSD ...)

3. **API, integraciones y conectores:** Se interactúa con Intalio|BPMS servidor utilizando un amplio servicio (SOAP) API Web, integración con diferente forma y tecnologías de interfaz de usuario, se pueden invocar servicios web SOAP y XML a través de REST usando arrastrar y soltar y se puede ejecutar consultas SQL y procedimientos almacenados y obtener los resultados en su proceso.

4. **Supervisión de la actividad:** Tiene un editor gráfico «Dashboard», métricas de negocio para definir los objetos de negocio y métricas que mejor reflejen sus necesidades empresariales y se utiliza una de las decenas de «widgets» para monitorear su negocio y aplicación.

5. **Reglas de Negocio:** Tiene BPM para modelar las reglas de negocio en un editor, Business Rule Engine (BRE) para Implementar y ejecutar las reglas de negocio, procesos y eficiencia de racionalización y se reutiliza las reglas de negocio de los procesos o de otros sistemas.

6. **Lista de tareas del usuario:** Acceso a la lista de tareas de la web o un dispositivo móvil, se puede gestión de ausencia en el trabajo, se pueden reclamar y revocar tareas, se puede guardar un borrador de las respuestas a la tarea y luego enviarlas, se puede exportar la lista de tareas en formato PDF o Excel y se puede adjuntar documentos a la tarea y almacenarlos de forma local o en el Document Management System (DMS) de su elección.

7. **Autenticación y autorización:** Basado en el control de acceso, directorio de acceso basado en archivos, integración con su autenticación de usuarios Lightweight Directory Access Protocol (LDAP) existentes y se puede examinar el LDAP directamente desde el diseñador y los usuarios arrastrar y soltar y los roles que los asocian con las tareas.

8. **Entornos de certificados, Portales y Gestión Documental:** Intalio|BPMS soporta Oracle, MSSQL, PostgreSQL, DB2, Sybase, MySQL y bases de datos, Internet Explorer, Firefox y Chrome, y cualquier sistema operativo. Se puede acceder a la lista de tareas desde un portlet Liferay y almacenar automáticamente archivos adjuntos de tareas en Alfresco.

9. **Rendimiento y escalabilidad:** Clústeres y equilibrio de carga, se puede asegurar de que fallos de hardware no afectan a su negocio y se realiza limpieza automática del servidor de datos antiguos para un máximo rendimiento.

10. **Implementación y administración de Procesos de Negocio:** Se puede desplegar un proceso con un solo clic, se puede ver versiones de proceso y elegir cuál debe ser activa, se puede definir los parámetros y criterios de valoración de implementación basados en cada uno de sus ambientes, se permite el registro de auditoría, se supervisan los procesos implementados y su estado de ejecución general y se pueden recuperar fallos de instancia con un solo clic.

11. **Idiomas:** Intalio|BPMS soporta Inglés, Francés, Japonés, Portugués y Español, y se puede traducir la plataforma a cualquier otro idioma que se desee.

12. **Soporte de productos y ayuda:** se puede acceder a documentación profesional, se pueden hacer preguntas ilimitadas al equipo de apoyo profesional, se obtiene apoyo de «Intalio Solution Experts» e instructores profesionales de todo el mundo, se obtiene apoyo del foro de la comunidad y se realizan actualizaciones automáticas.

Trabaja con el «standard» BPEL y se compila cada modelo desarrollado en un fichero de ese lenguaje. Para la integración con aplicaciones Java, Intalio es capaz de desplegar procesos y publicarlos los automáticamente usando Apache Axis2, que es un motor de servicios web que debe ser invocado desde cualquier código Java a través de una petición en una mensaje SOAP.

Las herramientas que componen el software de Intalio utilizado para el desarrollo de este proyecto, están disponibles para descargar en el sitio web de Intalio y son las siguientes:

1. **Intalio|Designer 6.5.1.** Intalio Designer es una herramienta para el modelado de un proceso de negocio con BPMN. Esta notación se puede transformar a BPEL por la herramienta de diseño. Cuando es realizado el diseño, éste es ejecutado por el servidor de Intalio. Si anteriormente se ha utilizado Eclipse, se familiarizará rápidamente con Intalio Designer porque utiliza componentes de los proyectos de la comunidad de Eclipse.

   Una vez dentro del «Designer» se cuenta con un modelador gráfico de procesos, utiliza modelado BPMN 2.0, con la capacidad de arrastrar componentes BPMN desde varias paletas para construir el proceso. Los ficheros WSDL, que describen los servicios web que serán invocados durante la ejecución de un proceso se pueden arrastrar y soltar para mapear los parámetros que se quiera mandar arrastrando el archivo WSDL al diagrama de proceso de negocios, archivo .bpm. Si hacen falta transformaciones, se pueden utilizar expresiones XPath.

   Además, se puede definir formularios mediante Intalio Ajax y XForms. Generarán formularios que se podrán usar en los procesos. Mediante estos formularios, los usuarios podrán interactuar con los procesos definidos. En este proyecto se ha hecho uso de XForms, es un formato XML diseñado por el W3C para poder definir interfaces de usuario, principalmente formularios web. El editor de formularios Xforms es un componente de BPMS Workflow embebido en Intalio BPMS Designer. Permite crear simplemente formularios XForms arrastrando componentes desde una paleta. Además, genera automáticamente el código, integrando los mismos con los procesos. Además, el editor permite exportar los formularios a ficheros de imagen, útiles para la documentación del proyecto.

2. **Intalio|BPMS 6.5.1 Servidor.** Intalio Server es una parte independiente de la herramienta Intalio. Puede ser utilizado para ejecutar los procesos modelados por la herramienta de diseño, así como la ejecución de otros procesos. El servidor puede ser integrado sin problemas con el «Designer» de forma tal que los procesos modelados por el «Designer» pueden exportarse directamente al «Server» mediante un clic. El servidor puede utilizarse también para ser usado sin el «Designer» para poner a punto un «framework» para la ejecución de procesos de negocio modelados con otras herramientas en lugar de Intalio Designer.

   Los usuarios iniciarán los procesos y llevarán a cabo sus tareas. Los usuarios podrán ver sus procesos, sus tareas y sus notificaciones. En la vista de procesos, se encontrarán todos los procesos que estén activos y se puedan lanzar. Tras lanzarse el proceso, empezará la ejecución. En la vista de tareas, se mostrarán las tareas que el usuario tenga que realizar. Aparecerán los formularios para las tareas que lo soliciten y además, se podrán gestionar las tareas a través de un panel de acciones, se puede saltar la tarea, rechazarla, etc. Por último, en la vista de notificaciones, aparecerán todas las notificaciones que se le hayan enviado al usuario.

En la figura C.3 se observa la interfaz de Intalio|Designer.

Intalio|BPMS se ha utilizado para desarrollo de las composiciones.

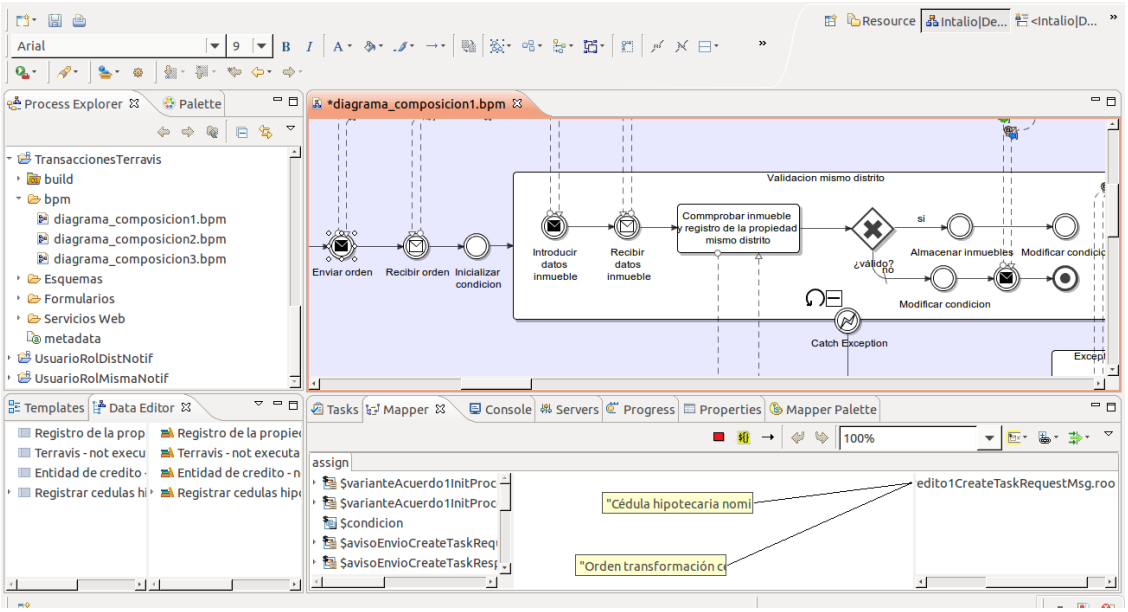


Figura C.3.: Interfaz de Intalio|Designer.

C.2.4. Apache ODE

Apache ODE [34] es una herramienta desarrollada por «Apache Software Fundation» bajo la licencia Apache, ejecuta procesos de negocio ejecutables basados en el estándar WS-BPEL 2.0. Se comunica principalmente con uno o más servicios web, enviar y recibir mensajes, la manipulación de datos y el manejo de excepciones (errores) según la definición de cualquier proceso dado. El motor es capaz de ejecutar tanto largos como cortos procesos vivos para coordinar todos los servicios que conforman un servicio o aplicación (orquestación). Sus características principales son:

1. Soporte del estándar WS-BPEL 2.0 de OASIS y el BPEL4WS 1.1.

2. Soporta dos capas de comunicación: una basada en Axis2 («Web Services http transport») y otro basado en el estándar Java Business Integration (JBI) (usando ServiceMix).

3. Soporte para HTTP WSDL Binding.

4. Alto nivel de API para el motor que le permite integrar el núcleo con prácticamente cualquier nivel de comunicación.

5. La compilación de WS-BPEL proporciona un análisis detallados y validación en la línea de comandos o en el despliegue.

6. Interfaz para la gestión de procesos, instancias y mensajes.

Intalio en su interior incorpora el motor Apache ODE.

C.3. Software utilizado para evaluación de la calidad

El objetivo principal que tiene este proyecto es que sea software de calidad. Con esta finalidad, se ha utilizado Sonar y Jenkins, se detallan a continuación.

C.3.1. Sonar

Sonar [27] es una plataforma que permite gestionar la calidad del código controlando los 7 ejes principales de dicha calidad del código:

- Arquitectura y diseño
- Duplicaciones
- Pruebas unitarias
- Complejidad
- Errores potenciales
- Reglas de codificación
- Comentarios

Es una plataforma para evaluar código fuente. Es software libre y realiza varios análisis de nuestro código usando diferentes herramientas (Checkstyle, PMD, FindBug y Clover) y presenta de manera unificada a través de su interfaz la información generada por ellas en forma de métricas que pueden ayudar a mejorar la calidad del código de un programa.

Se informa sobre código duplicado, estándares de codificación, porcentaje de cobertura con las pruebas unitarias así como los puntos para los que no se ha diseñado ninguna prueba, cobertura de código, complejidad ciclomática, posibles errores, porcentaje de comentarios, líneas duplicadas y diseño del software. El principal lenguaje soportado es Java, aunque, se puede extender a otros lenguajes. Más de 20 lenguajes de programación están cubiertos a través de «plugins» como Java, C #, C / C ++, PL / SQL, Cobol, ABAP, etc. Se integra con Maven, Ant y herramientas de integración continua como Atlassian Bamboo, Jenkins y Hudson.

En `https://neptuno.uca.es/sonar/` aparece una lista con todos los proyectos realizados por el grupo UCASE, entre los cuales, a este proyecto se le a asignado el nombre TransaccionesElectronicasGrode. Se accede al apartado específico de este pulsando sobre dicho nombre, aparece en pantalla el cuadro de mandos («dashboard»).

La interfaz web se divide en tres áreas:

1. **Barra superior.** Estés donde estés, puedes acceder a la barra superior:
   - **«Dashboards»** para volver a la página de inicio
   - **«Projects»** para acceder a la lista completa de los proyectos, vistas, desarrolladores o para acceder previamente a los proyectos visitados.
   - **«Measures»** para acceder al servicio de medidas.
   - **«Issues»** para acceder al servicio de conflictos.
   - **«Quality Profiles»** para examinar y gestionar los perfiles de calidad.
   - **«Settings»** para acceder a los ajustes del sistema (acceso restringido a los administradores de sistemas).
   - **«Log in» / <Name>** para iniciar la sesión.
   - **Búsqueda** de un componente: proyecto, archivo, vista, desarrollador, etc, para acceder rápidamente a él.

- **Ruta de navegación.** Esta ruta de navegación se puede hacer clic, por lo que podrá llegar rápidamente a un panel de control de nivel superior.
- **Estrella** para marcar el proyecto actual como favorito.
- **«Configuration»** para acceder a las páginas de configuración del proyecto actual.

2. **Menu izquierdo**. A la izquierda del todo, hay un cuadro azul donde aparece más información sobre el proyecto como:
   - **Puntos críticos («Hotspots»).** Se observa un listado de las principales métricas junto con un ranking de los cinco primeros.
   - **Revisiones («Reviews»).**
   - **Máquina del tiempo («Time machine»).** Se observa como a evolucionado la complejidad, el cumplimiento de las reglas y la cobertura del código a lo largo de la creación del proyecto. El «widget» de línea de tiempo ofrece la posibilidad de mostrar un gráfico que contiene datos históricos de hasta 3 métricas. Al pasar el ratón sobre la línea de tiempo mostrará los diferentes valores. El «widget» de tabla de histórico proporciona la capacidad para mostrar una tabla con los datos históricos para hasta 10 métricas.
   - **Conflictos («Issues»).** Cuando un componente no cumple con una regla de codificación, un conflicto se registra instantáneamente. Un conflicto se puede registrar en un archivo de origen o un archivo de prueba unitaria. Se muestran el número de conflictos existentes y se puede ver los detalles de cada uno.
   - **Componentes («Components»).** Se encuentran los diferentes módulos que componen el proyecto.
   - **Detalle de conflictos («Issues Drilldown»).** Se encuentran los niveles de gravedad y un listado con todos los paquetes de nuestro proyecto y, dentro de cada uno de ellos, las clases. Más abajo, se encuentran todas las reglas que se están incumpliendo, seguidos de su gravedad, una breve descripción y la clase donde esto ocurre.
   - **Diseño («Design»).** Se puede ver la relación entre las dependencias entre paquetes del proyecto.
   - **Bibliotecas («Libraries»).** Al tratarse de un proyecto java construido con Maven, aquí aparecen las dependencias con librerías externas del proyecto.
   - **Nubes («Clouds»).** Con la vista en forma de nubes de etiquetas se puede detectar puntos clave en nuestro proyecto.

     «Top risk». Donde el tamaño representa la complejidad media por método en la clase, y el color representa la cobertura de código o el nivel de cumplimiento de reglas.

     «Quick wins». Donde el tamaño representa las líneas de código de la clase. El color representa igualmente la cobertura o el nivel de cumplimiento.
   - **Comparar («Compare»).**

3. **Visualización de los datos.**

   El la figura C.4 se observa nuestro «dashboard» que ofrece una visión general de nuestro proyecto y su calidad. Se presentan los resultados de las métricas del proyecto de forma general. Arriba a la izquierda, informa de las líneas de código y clases, especificando, el número de líneas, las instrucciones y los ficheros, para las líneas de código y para las clases, el número de clases, métodos, paquetes y accesores. A la derecha, se puede ver los conflictos y cumplimiento de reglas o normas de estilo, y las clasifica en: bloqueadas, críticas, mayores, menores y de información. Abajo a la izquierda, se tiene una medición de la complejidad del código, tanto por métodos como por clases, y a la derecha, se informa si hay ciclos entre las dependencias. Abajo del todo se ve el porcentaje de cobertura del

código y resultado de las pruebas. Cada uno de todos estos resultados pueden consultarse para ver más detalles.

Una sección importante es "Conflictos" que indica los errores que tiene nuestro código dividido en niveles de gravedad clasificados anteriormente nombrados. Si se entra por ejemplo aquí, pulsando encima del porcentaje de las reglas de estilo cumplidas se aparecen dichos niveles de gravedad y se obtiene un listado con todos los paquetes de nuestro proyecto y, dentro de cada uno de ellos, las clases. Más abajo, se encuentran todas las reglas que se están incumpliendo, seguidos de su gravedad, una breve descripción y la clase donde esto ocurre. También, se puede haber pulsado los errores de un determinado nivel de gravedad en vez del porcentaje. Es muy útil para conocer que normas de estilo incumplidas, corregirlas y asegurar que nuestro código está escrito de acuerdo a las buenas prácticas de Java mejorando así en eficiencia, usabilidad y mantenibilidad, fundamentalmente.

Si se entra en el resultado de las pruebas, se observa un listado con todos los paquetes de nuestro proyecto y, dentro de cada uno, las clases con las distintas pruebas. En la figura C.5 se ven las pruebas para la clase ServicioHipotecaTest pasadas con éxito.

Las correcciones a destacar realizadas al proyecto tras ver los resultados de Sonar fueron:

- El nombre de variables y funciones deben coincidir con el patrón "^[a-z][a-zA-Z0-9]*$".
- Variables que debían ser privadas y tener métodos de acceso.
- Evite el uso de if ... else sin llaves.
- Este bloque de líneas comentadas con los de código debe ser eliminado.
- Nueva excepción en bloque catch, seguimiento de la pila original, se puede perder.
- Considere la posibilidad de devolver sencillamente el dato vs almacenarlo en variable local "registroPropiedad".
- Evite printStackTrace ().
- Boolean Expression Complexity. La complejidad de la expresión lógica es 5 (máxima permitida es 3).
- Class Type(Generic) Parameter Name. El nombre "Id" debe coincidir con el patrón "^[A-Z]$".
- Modificador "public" redundante.
- Evitar los campos privados no utilizados.
- No crear instancias de BigInteger y BigDecimal (ZERO, ONE, TEN) ya existente.
- Evitar duplicados literales (Avoid Duplicate Literals).
- No esta permitida la asignación del parámetro.
- Las clases de utilidad no deben tener un constructor público o por defecto.
- Cyclomatic Complexity. La complejidad ciclomática es 12 (máxima permitida es 10).

C.3.2. Jenkins

Jenkins [28] es un software de integración continua de código abierto. Construido con Java, que proporciona 970 «plugins» para apoyar la creación y el ensayo prácticamente cualquier proyecto. Surgió inicialmente como bifurcación de un proyecto llamado Hudson aunque muchos consideran que tan solo fue un cambio de nombre. Con esta herramienta se puede desplegar nuestro código en un servidor donde será compilado y testeado.

Es un sistema corriendo en un servidor que es un contenedor de servlets, como Apache Tomcat. Jenkins tiene soporte para herramientas de control de versiones, algunas como SVN, CVS, Git y y puede ejecutar proyectos basados en Apache Ant y Apache Maven. El desarrollador principal es Kohsuke Kawaguchi, que trabajaba en «Sun Microsystems». Liberado bajo licencia Massachusetts Institute of Technology (MIT).

Jenkins se encarga de monitorizar las ejecuciones de trabajos repetitivos, tales como la construcción de un proyecto de software o trabajos ejecutados por cron. Entre esas cosas, Jenkins actual se centra en los dos trabajos siguientes:

- **Construcción y pruebas de software continuas.** Jenkins ofrece una herramienta fácil de usar llamado sistema de integración continua, por lo que es más fácil para los desarrolladores integrar cambios en el proyecto, y lo que es más fácil para los usuarios obtener una construcción nueva de una forma simple. La construcción automatizada y continua incrementa la productividad.
- **Monitorización de la ejecución de tareas que se ejecutan externamente,** tales como las tareas cron y procmail; incluso aquellas que se ejecutan en máquinas remotas. Jenkins mantiene los resultados y hace que sea fácil darte cuenta cuando algo está mal.

Jenkins es fácil de instalar, la simplicidad de uso con una interfaz web simple proporciona: obtención de código de diversos tipos de repositorios como Subversion o CVS, compilación, ejecución de «test» (JUnit), presentación de reportes, automatización de comandos, notificaciones vía Instant Messaging (IM), e-mail y RSS, etc. Permite desplegarse en cualquier máquina o servidor y su uso y configuración, basada en formularios web permite su utilización desde cualquier dispositivo con navegador. A través de Jenkins se puede configurar la manera de analizar el código fuente de nuestro proyecto e integrar la herramienta con otras aplicaciones para hacer un análisis profundo de los resultados obtenidos. Facilita el seguimiento a procesos, encontrar defectos y la toma de decisiones sobre inconsistencias o mejoras en el software. Integra al equipo de trabajo, facilita la organización, estabilidad y calidad del proyecto. A través de «plugins» se amplía su funcionalidad y compatibilidad con un gran número de servicios, tecnologías y lenguajes que no son soportados inicialmente.

Se hacen estas pruebas tras la subida de cambios y así, proporcionar que el proyecto siempre funcione correctamente. En la figura C.6 se puede ver una de las ejecuciones del código.

C.4. Software utilizado para la edición de textos

C.4.1. kile 2.1

Kile [29] es un editor de TeX/LaTeX para el entorno de escritorio K Desktop Environment (KDE) fácil de usar. Se puede compilar, convertir y ver el documento con un solo clic, posee autocompletado de los comandos LaTeX, hace que cueste muy poco trabajo comenzar un nuevo documento con las plantillas y los asistentes, tiene fácil inserción de muchas etiquetas y símbolos estándar y la opción de definir (un número arbitrario de) etiquetas definidas por el usuario. Algunas características más son que la búsqueda la realiza hacia delante e inversamente, encontrar los capítulos o las secciones es muy fácil (Kile construye una lista de todos los capítulo, etc en el documento y puede utilizar la lista para saltar a la sección correspondiente), reúne los documentos que pertenecen a un mismo proyecto, es fácil la inserción de citas y referencias al

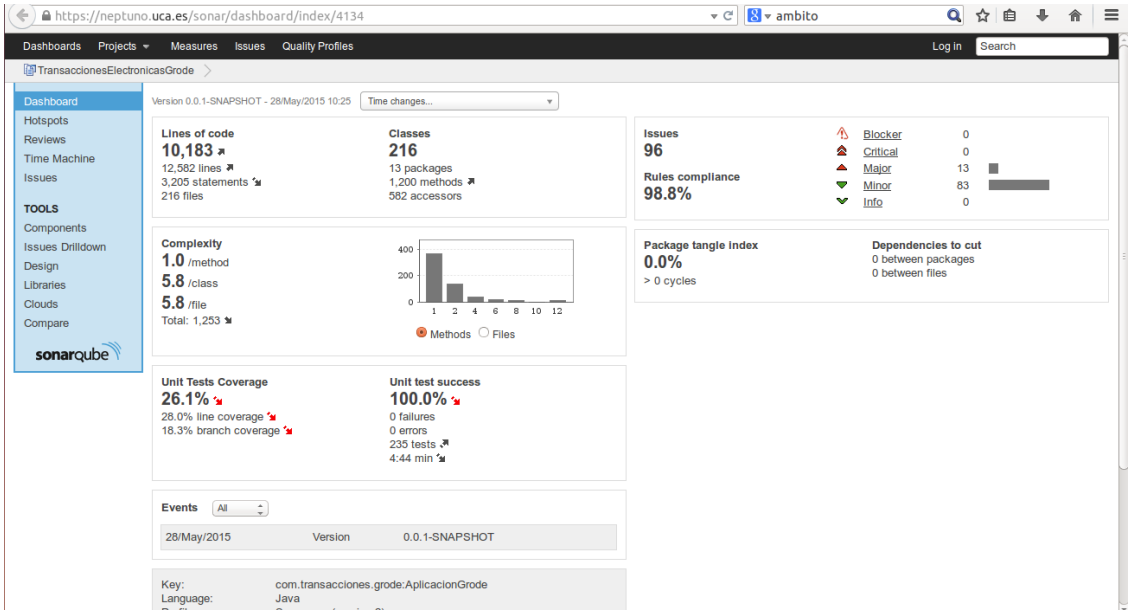


Figura C.4.: Pantalla principal de Sonar

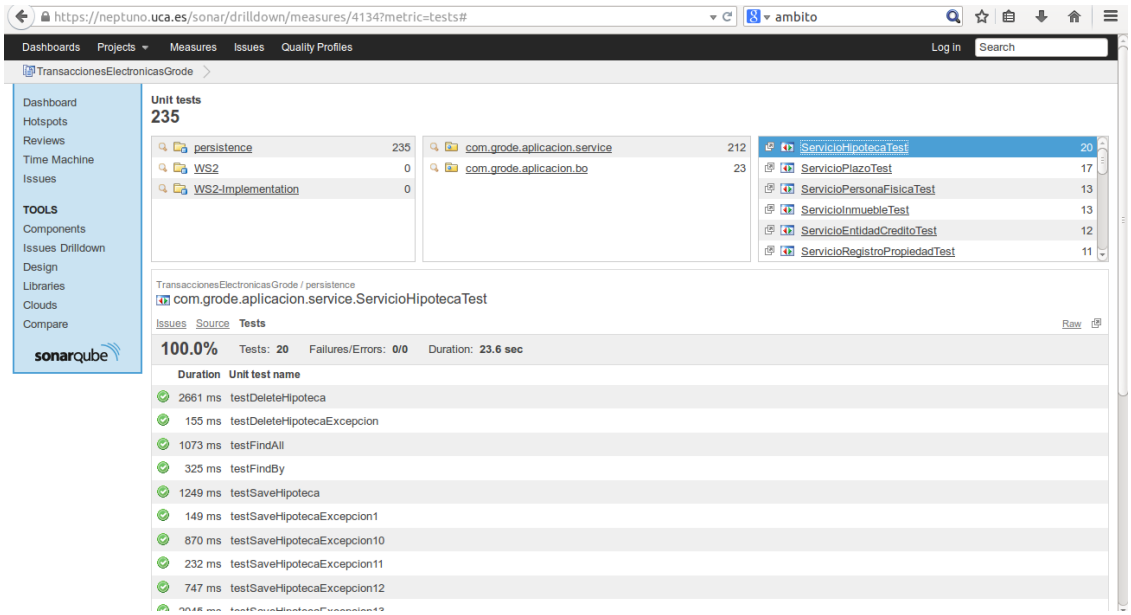


Figura C.5.: Sonar - Pruebas

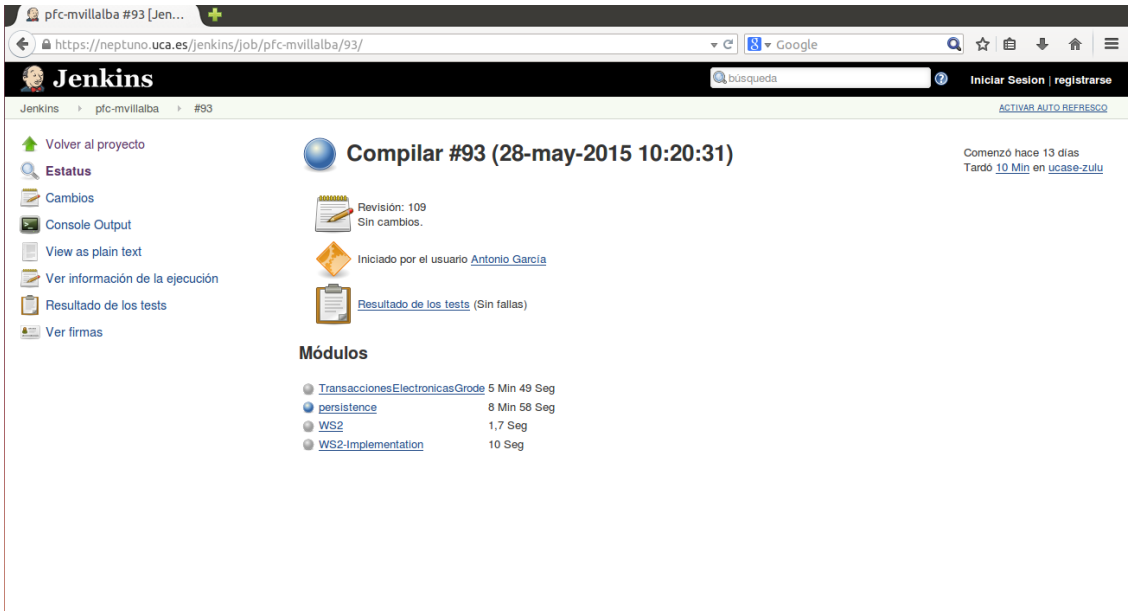


Figura C.6.: Pantalla de Jenkins

usar proyectos, posee un sistema de construcción flexible e inteligente para compilar sus documentos LaTeX, la vista previa es rápida, tiene un fácil acceso a las diversas fuentes de ayuda y finalmente, los comandos de edición son avanzados. En la figura C.7 se puede observar la interfaz de kile.

Kile se ha utilizado para escribir toda la documentación en LaTeX.

C.4.2. LATEX

LaTeX [30] es un software para documentos de composición tipográfica. En otras palabras, es un sistema de preparación de documentos. Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados con LaTeX es comparable a la de una editorial científica de primera línea. Es un software libre de código abierto y originalmente fue escrito por Leslie Lamport.

LaTeX utiliza extensamente archivos de estilo llamados clases y paquetes, por lo que es fácil de diseñar y modificar la apariencia del documento y todos sus detalles. LaTeX está disponible para casi todos los sistemas operativos, como Windows, Linux, Mac OS X, y muchos más. Genera salida PDF imprimible y de fácil lectura. Además de PDF, es compatible con DeVice-Independent (DVI), PostScript, y la salida HTML.

Con LaTeX, la elaboración del documento requiere normalmente de dos etapas: en la primera hay que crear mediante cualquier editor de texto llano un archivo o fichero fuente que, con las órdenes y comandos adecuados, contenga el texto que se quiera imprimir. La segunda etapa consiste en procesar este archivo; el procesador de textos interpreta las órdenes escritas en él y compila el documento, dejándolo preparado para que pueda ser enviado a la salida correspondiente, ya sea la pantalla o la impresora. Si se quiere añadir o cambiar algo en el documento, se deberán hacer los cambios en el archivo fuente y procesarlo de nuevo.

La documentación completa ha sido escrita en LaTeX.

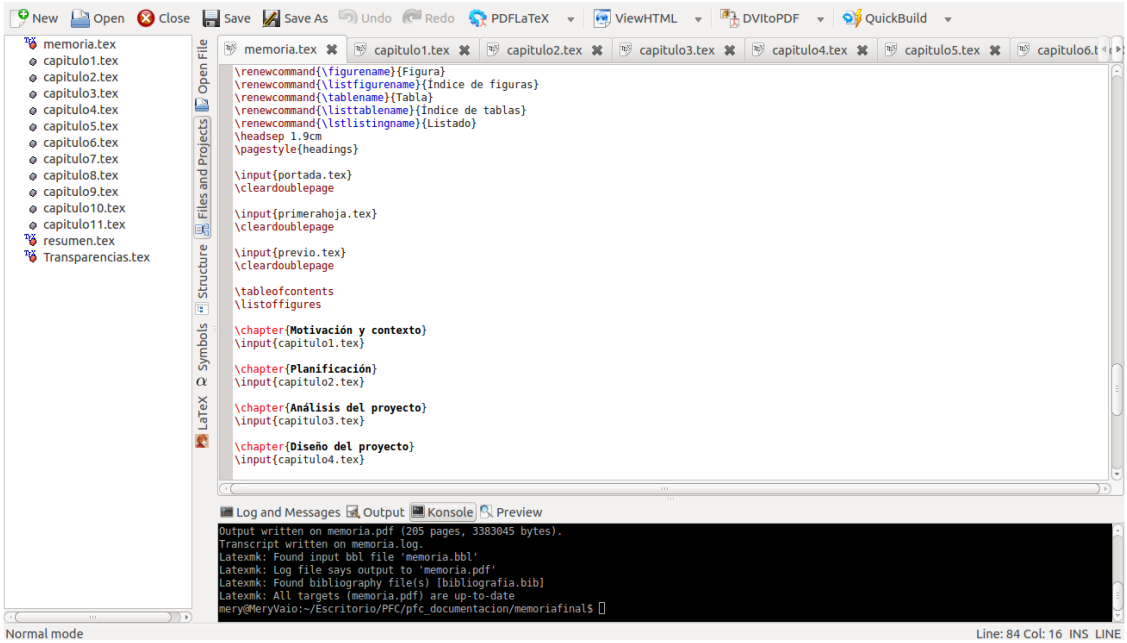


Figura C.7.: Interfaz del software Kile.

C.5. Otras tecnologías utiliazadas

C.5.1. Dia

Dia [31] es una herramienta bastante sencilla de utilizar con la que se puede crear diferentes tipos de diagramas, está inspirado más o menos por el programa comercial de Windows 'Visio', aunque más orientado hacia esquemas informales para uso ocasional. Se puede utilizar para dibujar diferentes tipos de diagramas. Actualmente cuenta con objetos especiales para ayudar a dibujar diferentes tipos de diagramas, como diagramas UML, diagramas de flujo, diagramas de entidad-relación, etc. Su interfaz es bastante accesible e intuitiva, consta de un panel de objetos, que dependiendo del tipo de diagrama que se escoja puede cambiar. Los objetos se arrastran al documento en el que se esté trabajando para formar el esquema que se desee. También es posible añadir soporte para nuevas formas escribiendo archivos XML simples, utilizando un subconjunto de SVG para dibujar la forma.

Además, ofrece una serie de herramientas que facilitan el desarrollo de cualquier diagrama, por ejemplo, reglas y una cuadrícula a la que se pueden ajustar los distintos objetos. Puede exportar diagramas a varios formatos, incluyendo Encapsulated PostScript (EPS), SVG, xfig, Windows Metafile Format (WMF) y Portable Network Graphics (PNG), e incluso a código LaTeX. También se pueden exportar los diagramas directamente mediante la generación de código C++, Java, Python y Pascal. En la figura C.8 se observa la interfaz de Dia.

Se ha utilizado este software para la creación de los diagramas de la documentación.

C.5.2. ArgoUml

ArgoUml [32] es la herramienta líder de modelado UML publicada bajo el código abierto Eclipse Public License e incluye soporte para todos los diagramas UML. Se ejecuta en cualquier plataforma Java y está disponible en diez idiomas. Ahora es el proyecto de código abierto organizado anualmente por Tigris.org. En la figura C.9 se observa la interfaz de ArgoUml.

Se ha utilizado este software para la creación del modelo conceptual de datos.

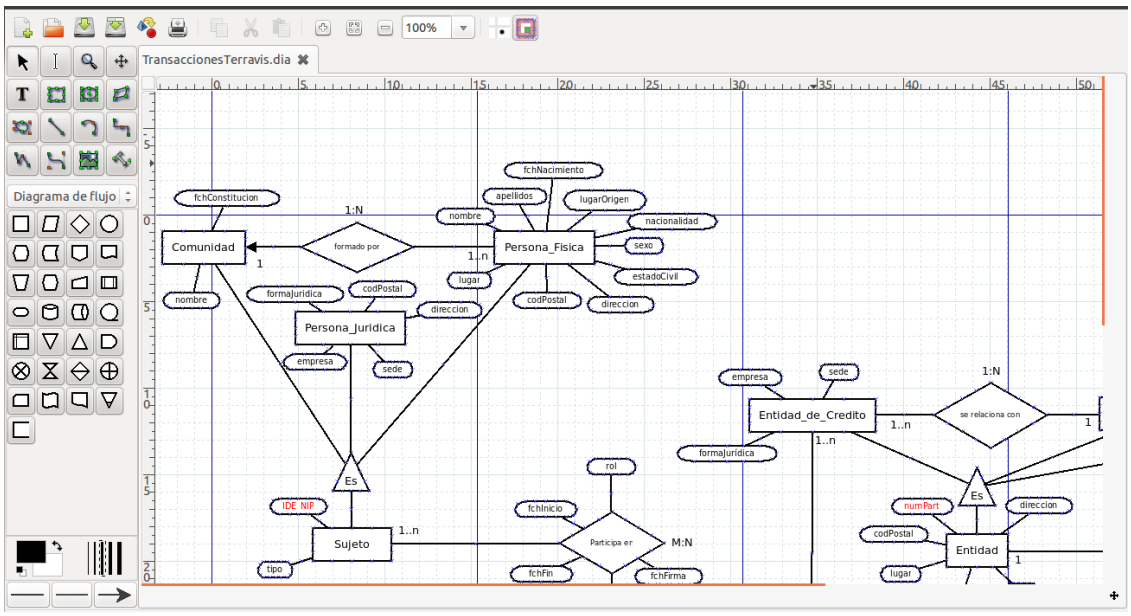


Figura C.8.: Interfaz del software Dia.

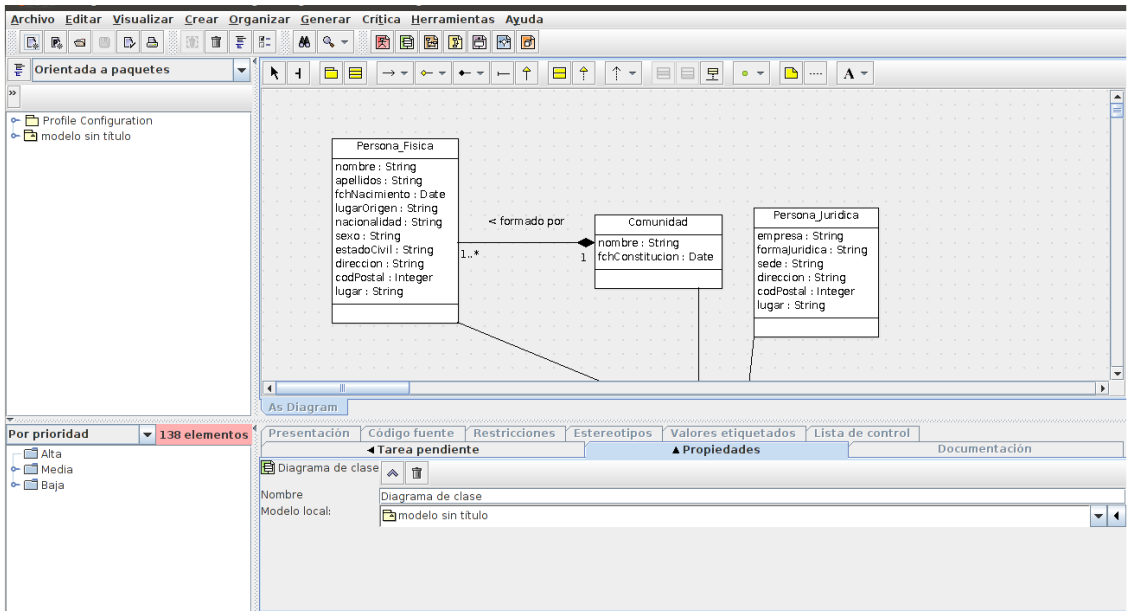


Figura C.9.: Interfaz del software ArgoUml.

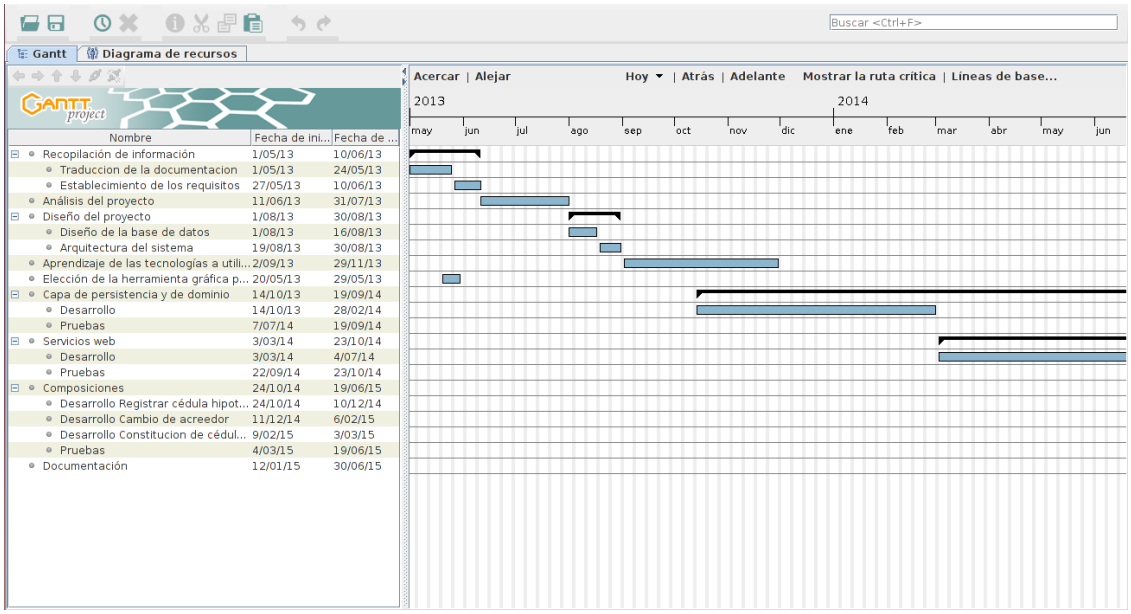


Figura C.10.: Interfaz del software GanttProject.

C.5.3. GanttProject

GanttProject [33] es una herramienta de escritorio multiplataforma para la programación y gestión de proyectos. Se ejecuta en Windows, Linux y MacOSX, es libre y su código es «opensource». Permite realizar diagramas de GANTT a fin de planificar un proyecto y gestionar los recursos. Es una herramienta completa con funcionalidades para importar y exportar hacia Microsoft Project y además permite exportar a los formatos Joint Photographic Experts Group (JPG), PNG, PDF y HTML. Es muy completa en cuanto admite la planificación de un proyecto de forma absolutamente visual. Un punto interesante es que permite establecer dependencias que interrelacionan las tareas, de tal forma que una tarea no podrá comenzar hasta que esté finalizada la anterior. En la figura C.10 se observa la interfaz de GanttProject.

Esta herramienta se ha utilizado para la creación del diagrama de Gantt que aparece en el capítulo 2 de esta documentación.

C.5.4. yEd

yEd [50] es un completo y práctico editor que genera rápidamente y eficazmente diagramas de alta calidad. Puedes trabajar con la paleta de símbolos, flechas y figuras que incorpora. Admite distintas estructuras, como jerárquico, orgánico, circular, en forma de árbol o por etiquetas, así como la posibilidad de agrupar elementos de manera automática o manual. Además, ofrece varios formatos de salida, además de la impresión.

Se ha utilizado este software para la creación de los diagramas de la documentación.

Lista de acrónimos

1FN Primera Forma Normal. 101, 102

2FN Segunda Forma Normal. 102

3FN Tercera Forma Normal. 102

4FN Cuarta Forma Normal. 103

5FN Quinta Forma Normal. 103

AOP Aspect Oriented Programming. 108

API Application Programming Interface. 73, 76, 105, 106, 191, 193

B2B Business to business. 33

BD Base de Datos. 50–52, 54–58, 120, 125, 126, 130, 141–153, 180

BO Business Object. 73, 137

BPEL Business Process Execution Language. 16, 20, 22, 28, 37, 38, 41–43, 74, 120, 126, 130, 155, 160, 184, 190, 192

BPM Business Process Management. 73, 74, 191

BPMN Business Process Modeling Notation. 11, 16, 27, 28, 33, 36, 42, 43, 75, 162, 184, 190–192

BPMS Business Process Management System. 42, 117, 163, 173, 178, 179, 182, 190–192

BPTS BPELUnit Test Suite. 16, 37, 38, 155, 160, 184, 186

BRE Business Rule Engine. 191

BSD Berkeley Software Distribution. 17

CDI Context and dependency injection. 190

COC Convention Over Configuration. 73, 109

CPU Central Processing Unit. 18

CRUD Create Read Update Delete. 43, 73, 78, 137, 140

CVS Concurrent Versions System. 17, 18, 189, 197

CXF Celtix/XFire. 115, 190

DAO Data Access Object. 73, 106–109, 111, 115, 137

DBA DataBase Administrator. 188

DER Diagrama de Entidad-Relación. 85

DIP Dependency Injection Principle. 73

DMS Document Management System. 191

DRY Don't Repeat Yourself. 73, 106

DSS Diagrama de Secuencia del Sistema. 59

DTD Document Type Definition. 26

DTO Data Transfer Objects. 115

DVI DeVice-Independent. 199

EJB Enterprise JavaBeans. 190

EPS Encapsulated PostScript. 200

ER Entidad-Relación. 79, 188

FLWOR For, Let, Where, Order by, Return. 118

FNBC Forma Normal de Boyce-Codd. 102

GPL GNU General Public License. 187

HTML HyperText Markup Language. 117, 199, 202

HTTP Hypertext Transfer Protocol. 25, 114, 115, 193

IBAN International Bank Account Number. 85, 94, 100, 101

IBM International Business Machines. 187, 189

IDE Integrated Development Environment. 178, 180, 188, 189

IEEE Institute of Electrical and Electronics Engineers. 133

IM Instant Messaging. 197

IOC Inversion of Control. 73

ISO International Standards Organization. 75

ISP Interface Segregation Principle. 73

IT Information Technology. 74, 75

J2EE Java 2 Enterprise Edition. 190

Java EE Java Platform Enterprise Edition. 16, 105, 178, 188, 189

JAX-RS Java API for RESTful Web Services. 190

JAX-WS Java API for XML Web Services. 115, 116, 190

JAXB Java Architecture for XML Binding. 115

JBI Java Business Integration. 193

JDBC Java Database Connectivity. 106, 179, 180

JDT Java Development Tools. 189

JEE Java Enterprise Edition. 12, 115, 178, 188, 189

JMS Java Message Service. 115, 190

JPA Java Persistence API. 105–107, 109, 153, 189, 190

JPG Joint Photographic Experts Group. 202

JSF JavaServer Faces. 189, 190

JSON JavaScript Object Notation. 115

JSP JavaServer Pages. 190

JSTL JSP Standard Tag Library. 190

JTA Java Transaction API. 190

KDE K Desktop Environment. 197

KPI key performance indicator. 74

LDAP Lightweight Directory Access Protocol. 191

MD5 Message-Digest Algorithm 5. 116

MIT Massachusetts Institute of Technology. 197

OASIS Organization for the Advancement of Structured Information Standards. 19, 193

ODE Orchestration Director Engine. 118, 160, 190, 193

ODF Open Document Format. 117

OMG Object Management Group. 27

ONU Organización de las Naciones Unidas. 33

ORM Object Relational Mapping. 105

PDF Portable Document Format. 57, 93, 172, 191, 199, 202

PFC Proyecto Fin de Carrera. 15–17, 41–44, 118, 161, 162, 187

PNG Portable Network Graphics. 200, 202

POM Project Object Model. 35

POO Programación Orientada a Objetos. 18

RBAC Role Based Access Control. 43

REST Representational State Transfer. 43, 114, 115, 191

RPC Remote Procedure Call. 24, 25

RSS Really Simple Syndication. 17, 197

SAAJ SOAP with Attachments API for Java. 115

SCM Software Configuration Management. 17

SHA Secure Hash Algorithm. 116

SOA Arquitectura Orientada a Servicios. 27, 71, 74, 76, 77, 113, 190

SOAP Simple Object Access Protocol. 16, 23–26, 42, 114–116, 162, 191, 192

SQL Structured Query Language. 118, 187, 188, 191

SSO Single Sign-On. 43

SVG Scalable Vector Graphics. 117, 200

SVN Subversion. 17, 18, 197

SWT Standard Widget Toolkit. 180

UDDI Universal Description, Discovery and Integration. 77

UML Lenguaje Unificado de Modelado. 48, 58, 200

URI Uniform Resource Identifier. 25

URL Uniform Resource Locator. 38, 114, 163, 173, 180, 182, 184

VCS Version control systems. 17

W3C World Wide Web Consortium. 23, 26, 27, 113, 117, 118, 192

WMF Windows Metafile Format. 200

WS-BPEL Web Services Business Process Execution Language. 15–17, 19–22, 28, 37, 41–43, 155, 161, 193

WSDL Web Services Description Language. 16, 19–24, 28, 38, 42, 77, 114–116, 155, 162, 191–193

XHTML Extensible Hypertext Markup Language. 117

XML eXtensible Markup Language. 16, 19, 21, 23–28, 37, 42, 105, 106, 114, 115, 117, 118, 154, 155, 162, 174, 184, 191, 192, 200

XSD XML Schema Definition. 20, 26, 42, 191

XSLT Extensible Stylesheet Language Transformations. 118

Bibliografía

[1] K. Pant y M. Juric. *Business Process Driven SOA using BPMN and BPEL*. Packt Publishing, 2008. ISBN 978-1-84719-146-5.

[2] T. Erl. *SOA: Principles of Service Design*. Prentice Hall, 2007. ISBN 978-0-13-234482-1.

[3] OASIS. Web Services Business Process Execution Language 2.0. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`, 2007. Última visita: 01-07-2015.

[4] BPELUnit. `http://bpelunit.github.io/`, 2014. Última visita: 01-07-2015.

[5] B. Eckel. *Piensa en Java*. Pearson Education, cuarta edición, 2007. ISBN 978-84-8966-034-2.

[6] I. Medina-Bulo, G. Aburruzaga-García y F. Palomo-Lozano. *Fundamentos de C++*. Servicio de publicaciones de la Universidad de Cádiz, segunda edición, 2009.

[7] S. S. Services. Terravis enables electronic business transactions between land registries, notary's offices and banks. `http://www.six-group.com/dam/about/downloads/media/media-releases/2012/media_release_201201260714_en.pdf`, 2012. Última visita: 01-07-2015.

[8] S. S. Services. Land register of Canton Berne activated in Terravis. `http://www.six-group.com/dam/about/downloads/media/media-releases/2013/0710-e-Terravis-Bern.pdf`, 2013. Última visita: 01-07-2015.

[9] S. Company. *Maven: The Definitive Guide*. O'Reilly Media, Inc., Septiembre 2008. ISBN 978-0-596-51733-5.

[10] Srirangan. *Apache Maven 3 Cookbook*. Packt Publishing, Agosto 2011. ISBN 978-1-84951-244-2.

[11] W3C. Web Services Description Language 2.0. `http://www.w3.org/TR/wsdl20`, Junio 2007. Última visita: 01-07-2015.

[12] DbUnit. `http://dbunit.sourceforge.net/`. Última visita: 01-07-2015.

[13] OMG. Documents Associated with Business Process Model and Notation (bpmn) Version 2.0. `http://www.omg.org/spec/BPMN/2.0`, 2011. Última visita: 01-07-2015.

[14] W3C. XML Schema. `http://www.w3.org/TR/xmlschema-2`, Octubre 2004. Última visita: 01-07-2015.

[15] W3C. Simple Object Access Protocol. `http://www.w3.org/TR/soap12-part1`, Abril 2007. Última visita: 01-07-2015.

[16] W3C. XPath. `http://www.w3.org/TR/xpath20`, Diciembre 2010. Última visita: 01-07-2015.

[17] W3C. XQuery. `http://www.w3.org/TR/xquery/`, Diciembre 2010. Última visita: 01-07-2015.

[18] Forja del Departamento de Lenguajes y Sistemas Informáticos. `https://neptuno.uca.es/redmine/`. Última visita: 01-07-2015.

[19] B. Collins-Sussman, B. W. Fitzpatrick y C. M. Pilato. Version Control with Subversion. `http://svnbook.red-bean.com/nightly/en/index.html`, 2013. Última visita: 01-07-2015.

[20] T. E. Foundation. ECLIPSE. `https://www.eclipse.org/`, Mayo 2013. Última visita: 01-07-2015.

[21] Subversive. `http://www.eclipse.org/subversive/`. Última visita: 01-07-2015.

[22] W3C. XQuery 1.0 and XPath 2.0 Functions and Operators. `http://www.w3.org/TR/xpath-functions/`, Diciembre 2010. Última visita: 01-07-2015.

[23] W3C. XQuery 1.0 and XPath 2.0 Data Model (XDM). `http://www.w3.org/TR/xpath-datamodel/`, Diciembre 2010. Última visita: 01-07-2015.

[24] I. Intalio. INTALIO. `http://www.intalio.com`, Mayo 2013. Última visita: 01-07-2015.

[25] JUnit. `http://junit.org/`. Última visita: 01-07-2015.

[26] O. C. and/or its affiliates. MySQL. `http://www.mysql.com/`, Mayo 2013. Última visita: 01-07-2015.

[27] S. S.A. Sonar. `http://www.sonarqube.org/`, 2013. Última visita: 01-07-2015.

[28] C. C. A. . U. license. Jenkins. `http://jenkins-ci.org/`. Última visita: 01-07-2015.

[29] Kile. `http://kile.sourceforge.net/`. Última visita: 01-07-2015.

[30] S. Kottwitz. *LaTeX*. Packt Publishing, Marzo 2011. ISBN 978-1-84719-986-7.

[31] A. Larsson. Dia. `http://projects.gnome.org/dia/`. Última visita: 01-07-2015.

[32] Tigris.org. ArgoUml. `http://argouml.tigris.org/`. Última visita: 01-07-2015.

[33] GanttProject. `http://www.ganttproject.biz/`. Última visita: 01-07-2015.

[34] A. S. Foundation. Apache ODE. `http://ode.apache.org/`, 2013. Última visita: 01-07-2015.

[35] G. LARMAN. UML Y PATRONES, 2002.

[36] G. Booch, J. Rumbaugh y I. Jacobson. *Unified Modeling Language User Guide*. Addison-Wesley Professional, segunda edición, 2005. ISBN 978-0-321-26797-9.

[37] L. Bass, P. Clements y R. Kazman. *Software Architecture in Practice*. Addison-Wesley Professional, tercera edición, 2012. ISBN 978-0-321-81573-6.

[38] P. T. Fisher y B. D. Murphy. *Spring Persistence with Hibernate*, capítulo Architecting Your Application with Spring, Hibernate, and Patterns, páginas 5–8. Apress, 2010.

[39] C. Álvarez Caules. *Arquitectura Java Sólida*. Sun Certifified Enterprise Architect, 2012. ISBN 978-1-291-16766-5.

[40] R. Elmasri y S. B. Navathe. Fundamentos de Sistemas de Bases de Datos, 2007.

[41] A. Silberschatz, H. F. Korth y S. Sudarshan. Fundamentos de Bases de Datos, 2006.

[42] T. J. Teorey, S. S. Lightstone, T. Nadeau y H. Jagadish. *Database Modeling and Design*. Morgan Kaufmann, 2011. ISBN 978-0-12-382020-4.

[43] W3C. Web services architecture. `http://www.w3.org/TR/ws-arch/`, 2004. Última visita: 01-07-2015.

[44] N. Balani y R. Hathi. *Apache CXF Web Service Development*. Packt Publishing, 2009. ISBN 978-1-847195-40-1.

[45] B. Alex, L. Taylor y R. Winch. Spring Security Reference. `http://docs.spring.io/spring-security/site/docs/3.2.4.CI-SNAPSHOT/reference/htmlsingle/#jc`, 2014. Última visita: 01-07-2015.

[46] W3C. XForms. `http://www.w3.org/TR/xforms11/`, Octubre 2009. Última visita: 01-07-2015.

[47] M. G. Piattini, J. A. calvo Manzano, J. Cervera y L. Fernández. *Análisis y Diseño de Aplicaciones Informáticas de Gestion: Una perspectiva de Ingenieria del Software*. RA-MA, 2003. ISBN 84-7897-587-X.

[48] The Apache Software Foundation. Apache software license version 2.0. `http://www.apache.org/licenses/LICENSE-2.0`, 2004. Última visita: 01-07-2015.

[49] The Apache Software Foundation. TomEE. `http://tomee.apache.org/`, Mayo 2013. Última visita: 01-07-2015.

[50] yWorks. yEd. `http://www.yworks.com/en/products/yfiles/yed/`. Última visita: 01-07-2015.